SPACE COMPUTER CORPORATION

(2)

SCC-R-124-1

OBJECT ACQUISITION AND TRACKING FOR SPACE-BASED SURVEILLANCE

Prepared for:

Scientific Officer
Office of Naval Research
Surveillance and Sensors Division
Applied Research and Technology Directorate
ATTN: Dr. Keith Bromley (Code 126)
800 North Quincy Street
Arlington, VA 22217-5000

Under:

Contract N00014-89-C-0015

2800 Olympic Boulevard • Suite 104 • Santa Monica, California 90404–4119
(310)829–7733 Fax (310)829–1694

UNCLASSIFIED
SECURITY CLASSIFICATION OF THIS PAGE

REPORT DOCUMENTATION PAGE

Form Approved
OMB No. 0704-0188

1a. REPORT SECURITY CLASSIFICATION UNCLASSIFIED	1b. RESTRICTIVE MARKINGS
2a. SECURITY CLASSIFICATION AUTHORITY	3. DISTRIBUTION/AVAILABILITY OF REPORT Unlimited
2b. DECLASSIFICATION/DOWNGRADING SCHEDULE	
4. PERFORMING ORGANIZATION REPORT NUMBER(S) SCC-R-124-1	5. MONITORING ORGANIZATION REPORT NUMBER(S)
6a. NAME OF PERFORMING ORGANIZATION Space Computer Corporation 6b. OFFICE SYMBOL (If applicable)	7a. NAME OF MONITORING ORGANIZATION Office of Naval Research, Code 126
6c. ADDRESS (City, State, and ZIP Code) 2800 Olympic Boulevard, Suite 104 Santa Monica, CA 90404-4119	7b. ADDRESS (City, State, and ZIP Code) 800 North Quincy Street Arlington, VA 22217-5000
8a. NAME OF FUNDING/SPONSORING ORGANIZATION Strategic Defense Initiative 8b. OFFICE SYMBOL (If applicable) IST	9. PROCUREMENT INSTRUMENT IDENTIFICATION NUMBER N00014-89-C-0015
8c. ADDRESS (City, State, and ZIP Code) The Pentagon Washington, DC 20301-7100	10. SOURCE OF FUNDING NUMBERS PROGRAM ELEMENT NO. / PROJECT NO. / TASK NO. / WORK UNIT ACCESSION NO.

11. TITLE (Include Security Classification)

Object Acquisition and Tracking for Space-Based Surveillance

12. PERSONAL AUTHOR(S)

13a. TYPE OF REPORT	13b. TIME COVERED	14. DATE OF REPORT (Year, Month, Day)	15. PAGE COUNT
Final	FROM 88DEC TO 90MAY	91NOV27	101

16. SUPPLEMENTARY NOTATION

Research supported by Strategic Defense Initiative/Innovative Science and Technology and Managed by Office of Naval Research

17. COSATI CODES		
FIELD	GROUP	SUB-GROUP

18. SUBJECT TERMS (Continue on reverse if necessary and identify by block number)
Signal Processing, Computer Architecture, Image Processing Parallel Processing, Algorithms, Image Registration, Clutter Suppression, Velocity Filtering

19. ABSTRACT (Continue on reverse if necessary and identify by block number)

The basic purpose of the research described here was to develop algorithms and demonstrate actual hardware for a new approach to the detection and tracking of moving targets using a passive IR or visual sensor. The new approach differs from the traditional approach of dividing the required processing into time-dependent, object-dependent, and data-dependent processing stages. Instead, it bases detection on multiple image frames, and, accordingly, requires a smaller signal-to-noise ratio. It is sometimes referred to as "track before detect," and can lead to a significant reduction in total system cost by allowing greater detection range for a single sensor, or by allowing the use of smaller sensor optics. Under this program we have:

Developed and demonstrated the required algorithms including precise frame registration, suppression of stationary clutter, and velocity filtering;
Designed and fabricated the SCC-100 programmable real-time processor to implement these and other signal and image-processing algorithms;
Programmed the algorithms to run in real-time on the SCC-100 processor, and demonstrated real-time operation using recorded imagery from actual sensors.

(Continued on reverse.)

20. DISTRIBUTION/AVAILABILITY OF ABSTRACT	21. ABSTRACT SECURITY CLASSIFICATION
☒ UNCLASSIFIED/UNLIMITED ☐ SAME AS RPT. ☐ DTIC USERS	Unclassified

22a. NAME OF RESPONSIBLE INDIVIDUAL	22b. TELEPHONE (Include Area Code)	22c. OFFICE SYMBOL
Dr. Keith Bromley	(619) 553-2535	Code 7601-T

19. ABSTRACT (continued)

The processor developed employs a one-dimensional array of identical processing nodes implemented with standard commercial components. A 20-node system has a peak throughput of 2 GFLOPS and incorporates 40 MBytes of memory. Each node occupies a single board in a 14-inch high cabinet, and 20 such boards can be mounted in one 19-inch wide rack. When implemented with hybrid wafer-scale interconnect (HWSI), a 2-GFLOPS processor can be packaged in 35 cubic inches, including radiation shielding.

CONTENTS

CONTENTS (Continued)

1.0 EXECUTIVE SUMMARY

This report presents the results of research carried out by Space Computer Corporation under the U. S. government's Small Business Innovation Research (SBIR) Program. The work was sponsored by the Strategic Defense Initiative Organization and managed by the Office of Naval Research under Contracts N00014-87-C-0801 (Phase I) and N00014-89-C-0015 (Phase II).

The basic purpose of this research was to develop and demonstrate a new approach to the detection of, and initiation of track on, moving targets using data from a passive IR or visual sensor. The approach we have developed differs in very significant ways from the traditional approach of dividing the required processing into time-dependent, object-dependent, and data-dependent processing stages. In that approach individual targets are first detected in individual image frames, and the detections are then assembled into tracks. That requires that the signal-to-noise ratio in *each* image frame be sufficient for fairly reliable target detection. In contrast, our approach bases *detection* of targets on multiple image frames, and, accordingly, requires a smaller signal-to-noise ratio. It is sometimes referred to as "track before detect," and can lead to a significant reduction in total system cost. For example, it can allow greater detection range for a single sensor, or it can allow the use of smaller sensor optics. Both the traditional and the "track before detect" approaches are applicable to systems using scanning sensors, as well as those which use staring sensors. The advantages of our approach can be summarized as:

a) It provides better detection for a given false alarm rate, and allows detection of very weak targets in clutter;
b) It is robust in its ability to handle large numbers of targets simultaneously with no significant increase in computational resources;
c) It is readily implementable using a parallel processing architecture.

Under this program we have:

a) Developed and demonstrated the required algorithms;
b) Designed and fabricated a programmable real-time processor which can implement these and other signal and image-processing algorithms;
c) Programmed the algorithms to run in real-time on our processor, and demonstrated real-time operation using recorded imagery from actual sensors.

This research effort has had two major thrusts. The first is algorithm development, combined with proof-of-concept computer simulations using real and synthetic sensor data. The approach which we have developed utilizes the following sequence of operations:

a) Precise frame registration (to an accuracy on the order of 0.01 pixel) of each image in the sequence using only the images themselves;

b) Suppression of stationary clutter in the registered images (this typically provides a signal-to-clutter ratio improvement of 30 dB or more);
c) Velocity filtering (or "3-D matched filtering") to improve the signal-to-noise ratio and to determine target velocity as well as target position.

Real-time operation of these algorithms requires throughputs comparable to that of a supercomputer. However, the applications which are of interest to us and to SDIO also require compact, light-weight hardware. Therefore, we were led to the second thrust of our research, which is the design and demonstration of a programmable, extremely high-performance signal processor which can be implemented with available components, and which can eventually be packaged using advanced interconnect technology to achieve the small size and weight required for SDI and other military applications. The processor we developed, which we call the SCC-100, employs a one-dimensional array of identical processing nodes implemented (initially) with standard commercial components. The peak throughput per node is 100 MFLOPS, and each node incorporates 2 MBytes of static RAM memory. Therefore, a 20-node system (for example) has a peak throughput of 2 GFLOPS and incorporates 40 MBytes of memory. In its basic form, using conventional printed circuit-board packaging, each node occupies a single board in a 14-inch high cabinet, and 20 such boards can be mounted in one 19-inch wide rack. When implemented with hybrid wafer-scale interconnect (HWSI), a 2-GFLOPS processor will be packaged in 35 cubic inches, including radiation shielding.

Military applications of our work include such programs such as Brilliant Eyes, Brilliant Pebbles, E^2I, EndoLEAP, and smart weapons. Commercial applications include video compression, computer vision, and some robotics applications.

It should be noted that we have already proceeded into Phase III of this program as defined by the SBIR Act of 1982. For example, we were awarded a contract by General Dynamics Corporation for the development of a 19-node commercial version of the SCC-100. The first model of this processor, which has a throughput of over 1 GFLOPS, was delivered to General Dynamics in early 1990. We are currently marketing the SCC-100 for both government and non-government applications. In addition, in 1990 we were awarded contract DACA76-90-C-0002 for DARPA (by the U. S. Army Engineer Topographic Labs) to miniaturize the SCC-100 for space applications. Finally, our work on moving-target detection has provided the basis for a new algorithm for full-motion video compression. Commercial applications for this new algorithm include high-definition television (HDTV), interactive video, and multimedia presentations.

2.0 BACKGROUND

In current approaches to passive IR sensor signal and data processing for surveillance and fire control systems, track initiation and scan-to-scan correlation are performed with classical "track-while-scan" association techniques. These approaches can encounter major difficulties when large numbers of objects such as booster fragments and kinetic kill debris are present in the field-of-view. The basic difficulty results from the sensitivity of classical track-association techniques to such conditions as merging or crossing tracks, temporary loss of input data, poor background rejection, etc. The resulting track mis-associations cause errors in the estimated position and velocity of targets.

The difficulties caused by track mis-association can be alleviated by the use of velocity filtering, or "track before detect," techniques. These techniques, pioneered by Space Computer Corporation, provide for computationally-efficient track association combined with signal-to-noise ratio enhancement. The algorithms are extremely robust with respect to such conditions as merging or crossing tracks, loss of input data, etc., even with very large numbers of objects, and should reduce discrimination errors due to track mis-association to a low level. Velocity filtering also provides background suppression and may permit automatic rejection of debris resulting from kinetic kills.

During Phase I of this program we developed algorithms for object discrimination, background suppression, track association and kinetic kill debris rejection based upon velocity-filtering concepts. We also evaluated alternative processor architectures capable of executing these and other required algorithms. Architectures investigated included a wide variety of parallel machines, both special- and general-purpose in nature, and encompassed both the time-dependent and object-dependent portions of the overall processing chain. We concluded that a medium- to fine-grained programmable, parallel processor architecture was the most attractive from the viewpoints of flexibility, long-term reliability and performance. After further work during Phase II we settled on the medium-grained, pipeline architecture described in section 5.0 of this report for the design, fabrication, and demonstration carried out under Phase II of this program.

3.0 TECHNICAL OBJECTIVES

The technical objectives of this program were:

(a) To develop and evaluate processing algorithms for acquisition, tracking, and discrimination of targets of interest for strategic defense;
(b) To assemble a demonstration brassboard signal processor capable of executing representative algorithms in real-time;
(c) To program the algorithms for execution on the processor and perform a functional demonstration.

All of these objectives were met and, to a large extent, exceeded. The algorithm work is described in section 4.0 of this report; the signal processor hardware is described in section 5.0, and the programming and functional demonstration are described in section 6.0.

4.0 ALGORITHM DEVELOPMENT

During Phase II of this effort the emphasis of the SDIO architecture changed in response to evolving system requirements, and the emphasis of this effort changed accordingly. This change meant that less emphasis was placed on tracking and discriminating *very* large numbers of objects (the BSTS/SSTS problem), and more emphasis was placed on weak object detection (the Brilliant Pebbles/Brilliant Eyes problem).

4.1 Signal Processing Chain

The basic block diagram for the image processing to be discussed in this report is shown in Figure 1. Since here the discrimination of targets from background is based on the fact that targets will be seen to move against the stationary background, in that figure we assume that the background is in fact stationary in *sensor* coordinates, i.e., that the sensor output is not corrupted by jitter or other sensor motion. This can be accomplished by either mechanical or electronic means. Mechanical image stabilization requires that the actual sensor pointing be controlled to a small fraction of the instantaneous filed of view (IFOV), and is not usually practical. For example, the IFOV might be as small as 100 μrad, and stabilization should usually be to better than 0.01 pixels, or, in this case, to better than 10 μrad, which is better that the sensor stability that can be reasonably expected for most systems. Thus, we assume that electronic image stabilization will be required. This means that before the images are processed for target detection *per se*, each input image frame is compared to a reference frame, the offsets between the two frames are measured, and the input image frame is resampled so that its new "pixels" precisely correspond to those of the reference frame. The details of this process are described in section 4.2 of this report.

The next processing step is the suppression of all stationary clutter in the input images. This is accomplished by computing the *temporal* mean and variance for each pixel. In other words, two new image frames are computed from a sequence of input image frames. The first new image is the pixel-by-pixel temporal mean for the sequence of input frames, and the second new image is the pixel-by-pixel temporal standard deviation (or variance) for the sequence of input image frames. The clutter is then suppressed by subtracting the mean image from each input image on a pixel-by-pixel basis, and then dividing the difference by the standard deviation (or variance) on a pixel-by-pixel basis. As will be discussed in the following sections of this report, it is not always necessary to divide by the standard deviation or variance. For example, if the output of the imaging sensor is dominated by sensor noise which is the same for each of its detectors, then the standard deviation is practically the same for each pixel, and the division is not necessary. On the other hand, if the output is dominated by photon noise from the scene itself, then the temporal variations in each pixel tend to a Poisson distribution with the standard deviation equal to the mean intensity, so that pixels which contain bright objects have a higher standard deviation than do pixels which contain less bright objects. In that case division by the standard deviation (or variance) is desirable. (The processing and performance differences for division by the standard deviation and division by the variance are discussed in section 4.3.)

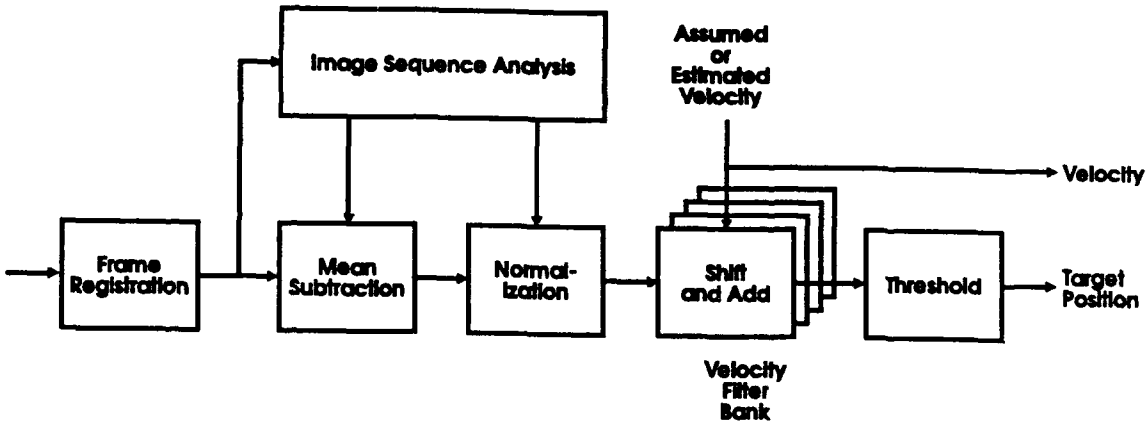


Figure 1. Target Acquisition Algorithm Chain.

Once the input images have been "whitened" by the suppression of stationary clutter, the temporal variation in each image pixel has unit variance and zero mean, because even those pixels which contain targets contain them for only a small fraction of the time. This processing step can improve the target-to-clutter ratio by up to 30 dB, and for some applications this is adequate to allow targets of interest to be detected without further processing. However, when that is not the case, even further improvement in the target-to-clutter ratio can be achieved by velocity filtering.

Velocity filtering is analogous to time delay and integrate (TDI) used with scanning multi-column sensors, except that it is applied to multiple frames collected many frame intervals apart in time. In the simplest case it consists of first two-dimensional shifting each frame with respect to its predecessors by an amount which just compensates for the target motion from frame to frame, so that in the shifted frames any target remains stationary. Then the shifted frames are added (integrated over time) on a pixel-by-pixel basis, which produces one new frame which we call the velocity filter output. This process increases the target-to-clutter ratio by approximately the square root of the number of frames added, and thus improves the target detectability. Of course, this procedure can not be carried out without *a priori* knowledge of the target velocity in sensor coordinates, and this velocity is usually not known before target detection and tracking. Therefore, in reality it is necessary to use a bank of velocity filters, with one filter matched to each region of the two-dimensional velocity space in which targets could appear. The design of such filter banks is described in section 4.4.

The final step in the processing chain is target detection, and this consists of comparing the output in each pixel of each velocity filter output with a threshold. The value of the threshold can be set according to the false-detection probability which can be tolerated, and it determines the probability of detection of a target with any specified amplitude. When the threshold is exceeded, the pixel which exceeds the threshold determines the position (in sensor coordinates) of the detected target, and the velocity filter output for which the threshold is exceeded determines the two-dimensional velocity (in sensor coordinates) of the target. At that point, track initiation is complete.

4.2 Frame Registration

Here we will describe a practical but effective phase correlation approach for measuring the 2-D translational offset between a pair of discrete image frames. The algorithm computes a 2-D cross-correlation of "phase-only" (or nonlinearly whitened) images derived from the original frames, and measures the location of the resulting correlation peak to sub-pixel accuracy using a simple parabolic interpolation scheme. The resulting jitter estimates are shown to be biased, but the bias is a function of signal processing parameters which do not depend on the spatial frequency content of the underlying scene itself. The bias can therefore be predicted in advance and corrected in real-time by means of a table-lookup.

A phase correlation procedure with peak interpolation and bias correction can provide very accurate sub-pixel frame registration measurements without the use of computationally expensive interpolation methods or prior knowledge of scene characteristics. The technique can be used to register frames that are subject to multiple-pixel as well as sub-pixel jitter or drift. Simulation experiments with

synthetically shifted frames indicate that the jitter measurement error for a typical clutter-limited IR scene is on the order of 0.01 pixel or less.

A key real-time implementation issue is whether the required discrete correlations may be performed in a circular fashion (with an FFT size equal to the frame dimensions) in order to save computation. An experiment performed on an actual sequence of IRMS long-wave IR sensor images with significant multiple-pixel drift suggests that circular processing is acceptable so long as the frame displacements are not a significant fraction of the total frame dimension.

4.2.1 Discrete Cross-Correlation. We begin by briefly reviewing the indirect (or frequency domain) method for computing the generalized cross-correlation between two sampled images [1,2]. Denote a pair of discrete 2-D image frames by $f_1(k,\ell)$ and $f_2(k,\ell)$ for pixel indices $k=0,...,M-1$ and $\ell=0,...,M-1$. The $N \times N$ discrete Fourier transforms of the frames are defined by

$$F_i(m,n) = \sum_{k=0}^{M-1} \sum_{\ell=0}^{M-1} f_i(k,\ell) e^{-j2\pi(km+\ell n)/N} \tag{1}$$

for frames $i=1,2$ and discrete spatial frequency components $m=-N/2,...,(N/2)-1$ and $n=-N/2,...,(N/2)-1$. We will allow $N \geq M$ to account for the possibility of frame zerofill prior to the 2-D FFT that implements (1). Zerofill is often used to avoid circular wraparound when discrete correlations are computed with the FFT method. The maximum non-circular correlation lag for an FFT size of N is N-M.

The conventional estimate of the cross-correlation between f_1 and f_2 is calculated by taking the inverse Fourier transform of a weighted cross-spectrum

$$G_w(m,n) = w(m,n)F_1^*(m,n)F_2(m,n) \tag{2}$$

where * denotes the complex conjugate and where w(m,n) is a 2-D weighting function that is used to prefilter the cross-spectrum and adjust the shape of the correlation peak. The cross-correlation estimate itself is defined for various 2-D integer lags (p,q) by

$$r(p,q) = \frac{1}{M^2} \sum_{m=-(N/2)+1}^{(N/2)-1} \sum_{n=-(N/2)+1}^{(N/2)-1} G_w(m,n) e^{j2\pi(pm+qn)/N} . \tag{3}$$

Note that the summations are performed symmetrically about the dc spatial frequency component located at (m,n)=(0,0).

The cross-correlation samples in (3) can be found by brute-force calculation or by taking an $N \times N$ inverse FFT of $G_w(m,n)$. The FFT approach is most efficient unless the required number of lags is known

to be fairly small. It provides circular cross-correlation estimates for pixel lags ranging from -N/2 to (N/2)-1 in either spatial dimension.

4.2.2 Discrete Phase Correlation. The discrete phase correlation is an alternative correlation function based on a cross-spectrum which is normalized to unit magnitude at all frequency components. With this approach, (3) is replaced by

$$\bar{G}_w(m,n) = w(m,n)\frac{F_1^*(m,n)}{|F_1(m,n)|}\cdot\frac{F_2(m,n)}{|F_2(m,n)|} = w(m,n)\frac{F_1^*(m,n)F_2(m,n)}{|F_1^*(m,n)F_2(m,n)|} \tag{4}$$

and the cross-correlation estimate becomes

$$\bar{r}(p,q) = \frac{1}{M^2}\sum_{m=-(N/2)+1}^{(N/2)-1}\sum_{n=-(N/2)+1}^{(N/2)-1}\bar{G}_w(m,n)e^{j2\pi(pm+qn)/N}. \tag{5}$$

A block diagram of the discrete phase correlation processing is shown in Figure 2. Mean removal pre-processing is included to eliminate the triangular "pedestal" that results from cross-correlating a pair of frames with nonzero average levels. Also, since the indices for FFT algorithms are conventionally defined as running from 0,...,(N/2)-1 rather than -(N/2)+1,....(N/2)-1 as shown in (5), a final (optional) 2-D circular shift is included to center the (0,0)-lag in the output cross-correlation array $\bar{r}(p,q)$.

Phase correlation is best viewed as a weighted cross-correlation of two "whitened" image frames whose spatial frequency components are defined by $F_1(m,n)/|F_1(m,n)|$ and $F_2(m,n)/|F_2(m,n)|$, respectively. A whitened frame retains just the phase information in the original spatial frequency components, and is often referred to as a phase-only image. Spectral whitening is a nonlinear filtering process which tends to enhance the high-frequency scene features (such as clutter edges) that contain most of the useful information for frame registration. Figure 3(a) shows an IRMS sensor frame collected in the longwave IR band (8-12 μm) which contains a relatively low-contrast desert terrain and sky clutter background. The phase-only image corresponding to this frame is shown in Figure 3(b). The whitening process emphasizes the relatively few significant clutter boundaries that are present in the original scene.

Since many natural scenes in the emissive IR region have a substantial amount of power concentrated at the low spatial frequencies, a conventional frame cross-correlation often produces a relatively broad output peak. The spectral whitening used by the phase correlation procedure tends to sharpen the peak, leading to more precise measurements of frame displacement. For example, Figure 4(a) shows a contour plot of the correlation function for a pair of jittered IRMS frames in the vicinity of the peak. A corresponding phase correlation function is shown in Figure 4(b).

4.2.3 Parabolic Peak Interpolation. The location of the peak in the phase correlation function (5) is a measure of the relative displacement or misregistration between the frames f_1 and f_2. To estimate the displacement to sub-pixel accuracy, the peak position must be accurately interpolated from the finite set of

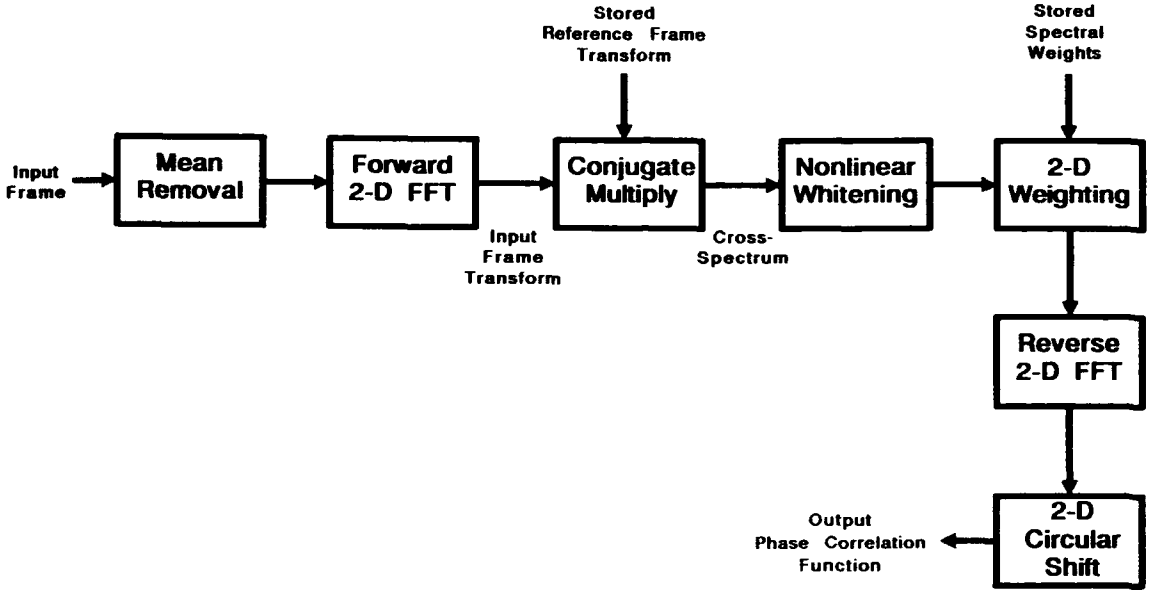
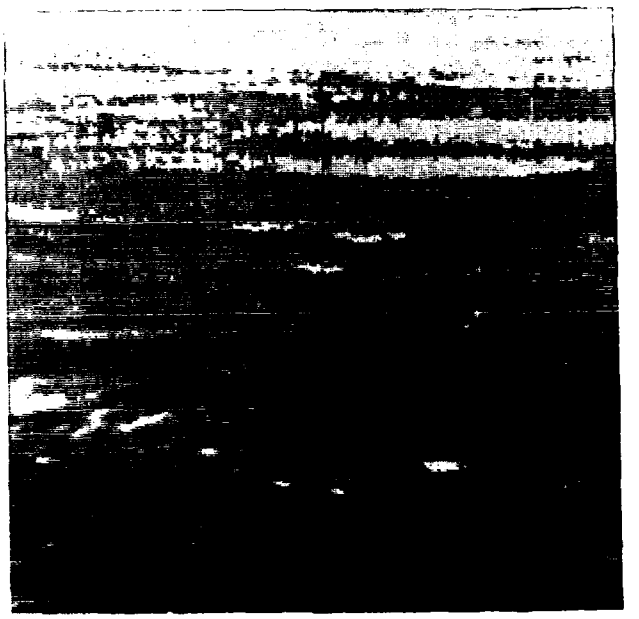
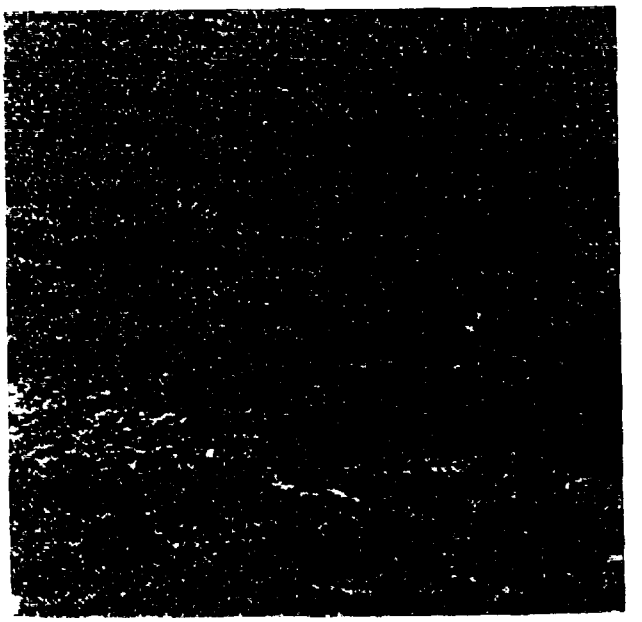


Figure 2. Block Diagram of Phase-Correlation Processing.

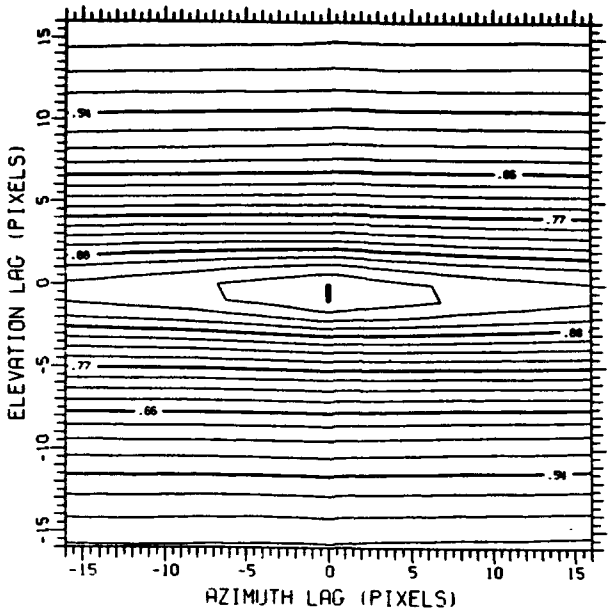


(a) Before Whitening.

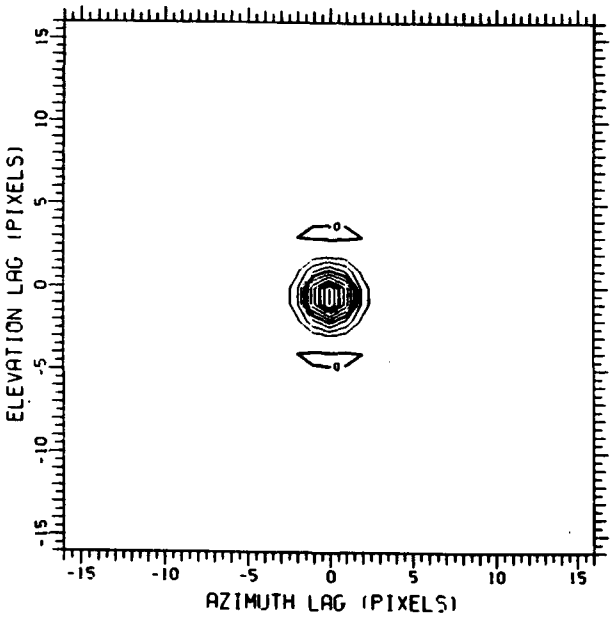


(b) After Whitening.

Figure 3. Long-Wave IRMS IR-Sensor Image: (a) Original Scene, and (b) After Nonlinear Whitening.



(a) Before Whitening.



(b) After Whitening.

Figure 4. Contour Plots of Cross-Correlation Functions for Images of Figure 3: (a) Original Scene, and (b) After Nonlinear Whitening.

samples. The classical solution to this problem involves the application of an appropriate 2-D interpolation function on the discrete cross-correlation estimate. For example, in the case of an unweighted correlation where w(m,n)=1 at all spatial frequencies, the desired interpolation function is $(\sin\pi m/\pi m)(\sin\pi n/\pi n)$. Unfortunately, the ideal interpolator can only be approximated on a finite-length record and is too computationally demanding to consider for most real-time applications.

A more practical approach is to fit a polynomial to the cross-correlation samples in the vicinity of the highest peak. A separable quadratic polynomial is the simplest choice since it uses only three points in either dimension: the peak sample and its neighbors to either side. Suppose the largest value of the phase correlation $\overline{r}(p,q)$ occurs at lag sample (p_o,q_o). Parabolic estimates of the true pixel misregistration $(\hat{p},\hat{q})$ are then given by

$$\hat{p} = p_o + \hat{\delta}, \tag{6a}$$

$$\hat{q} = q_o + \hat{\varepsilon}, \tag{6b}$$

where $\hat{\delta}$ and $\hat{\varepsilon}$ denote the estimates of the fractional shift in either frame dimension, and are defined by

$$\hat{\delta} = \frac{\overline{r}(p_o+1,\ q_o) - \overline{r}(p_o-1,\ q_o)}{2\left[-\overline{r}(p_o+1,\ q_o) + 2\overline{r}(p_o,\ q_o) - \overline{r}(p_o-1,\ q_o)\right]} \tag{7a}$$

$$\hat{\varepsilon} = \frac{\overline{r}(p_o,\ q_o+1) - \overline{r}(p_o,\ q_o-1)}{2\left[-\overline{r}(p_o,\ q_o+1) + 2\,\overline{r}(p_o,\ q_o) - \overline{r}(p_o,\ q_o-1)\right]}. \tag{7b}$$

For the phase correlation function $\overline{r}(p,q)$ in (4)-(5), $(\hat{p},\hat{q})$ correspond to the measured 2-D misregistration (in pixels) of the second frame f_2 with respect to the first frame f_1.

4.2.4 Measurement Bias Correction. In general, the misregistration measurements defined by (6) are biased. We have calculated the bias by assuming that the second frame appears as a translated replica of the first frame due to jitter and/or drift of the sensor boresight. The actual offset of frame f_2 with respect to frame f_1 in pixel units is defined as $(p_o+\delta,q_o+\varepsilon)$, where (p_o,q_o) and (δ,ε) represent the integer and fractional parts of the 2-D shift, respectively. Both δ and ε can take on values in the range [-0.5,+0.5].

Assuming that the highest peak of the phase correlation function occurs at the correct integer lag (p_o,q_o), the error in measuring the true shift is equivalent to error incurred in measuring its fractional components (δ,ε). For circular phase correlation combined with the parabolic peak estimators defined in (7), an exact analytical relationship between the measurement $\hat{\delta}$ (or $\hat{\varepsilon}$) and the actual value of δ (or ε) can be found. This relationship is

$$\hat{\delta}(\delta) = \frac{\sum w(m)\sin\left[\frac{2\pi m}{N}\right]\sin\left[\frac{2\pi m\delta}{N}\right]}{\sum w(m)\left[2-2\cos\left[\frac{2\pi m}{N}\right]\right]\cos\left[\frac{2\pi m\delta}{N}\right]} \tag{8}$$

where the indicated summations are performed over the spatial frequencies indices m = -N/2+1,...,N/2-1. The expression for $\hat{\varepsilon}(\varepsilon)$ has the same form; it is obtained simply by replacing δ with ε on the right side of (8).

A key assumption in the derivation of (8) is that the 2-D weighting function w(m,n) applied to the whitened cross-spectrum in (4) is real-valued, separable as w(m)w(n), and symmetric about dc spatial frequency. Under these conditions, the measurement $\hat{\delta}$ defined by (7a) depends on δ but not on ε, and likewise for $\hat{\varepsilon}$. This is convenient since it allows the measurement bias to be calculated independently in each dimension.

Figure 5 shows curves of $\hat{\delta}(\delta)$ or $\hat{\varepsilon}(\varepsilon)$ for N=256; Figure 5(a) for the case of uniform spectral weighting and Figure 5(b) for a heavily tapered weighting function defined by the square of the Hanning window:

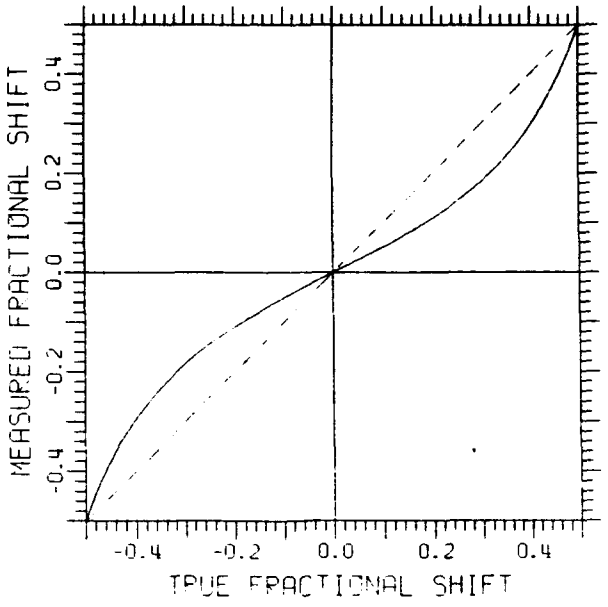
$$w(m) = \left[1+\cos\left[\frac{2\pi m}{N}\right]\right]^2. \tag{9}$$

The difference between the solid curve and the 45^{o} dotted line in each plot is the measurement bias. The magnitude of the bias is smaller for the narrowband Hanning^2 spectral window because the correspondingly broader correlation peak can be more accurately described by a 2nd-order polynomial.

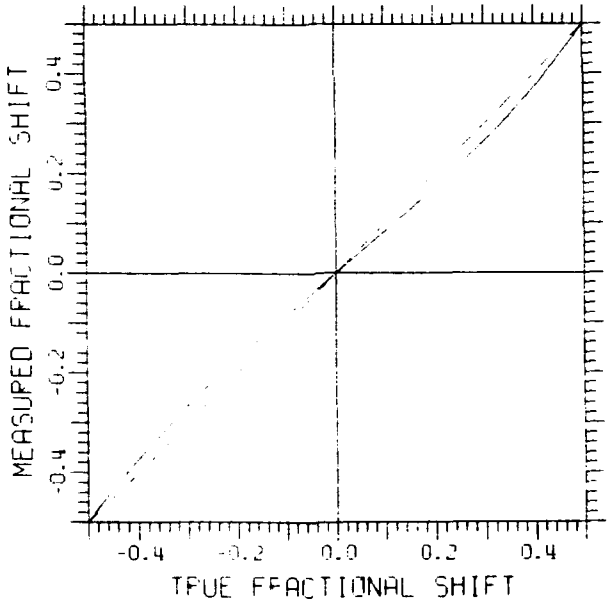
Figure 5 also shows that an invertible one-to-one relationship exists between the measurement $\hat{\delta}$ and the true fractional shift δ. This means that it is possible to correct the measurement bias using a lookup-table containing δ as a function of $\hat{\delta}$. Such a table can be pre-calculated and stored for rapid access in real-time. A complete block diagram of the proposed frame registration algorithm, including 2-D peak interpolation and a built-in bias correction step, is shown in Figure 6. This processing has been implemented in VAX-FORTRAN for processing in non-real-time.

It is important to note that the sub-pixel jitter measurements $(\hat{\delta},\hat{\varepsilon})$ depend only on the true fractional shifts (δ,ε) and the processing parameters N and w(n). They specifically do not depend on the characteristics of the background scene in the two frames. This feature is a direct result of the nonlinear whitening operation (4), and is an important practical advantage of phase correlation over conventional correlation. Since the relationships $\hat{\delta}(\delta)$ and $\hat{\varepsilon}(\varepsilon)$ are invertible, and since we can choose the transform dimension N and the spectral window functions w(m) and w(n), the registration scheme shown in Figure 6 can obtain unbiased jitter measurements without prior information about the background. Accurate measurements are more difficult to achieve with a conventional correlator, where the inherent bias of a finite-sample interpolator depends explicitly on the unknown spectrum of the scene (see, for example, [3]).

4.2.5 Circular vs. Non-Circular Correlation. The measurement equation (8) is exact when the sample cross-spectrum length N in either dimension is the same as the frame dimension M. This corresponds to the case where no zerofill is applied to the frames and the resulting 2-D phase correlation is circular. Circular correlation is the natural approach if the total frame misregistration is known to be less than a pixel or so in either dimension. It has the advantage of minimizing the required FFT size, and it is



(a) Uniform Cross-Spectrum Weighting.



(b) Hanning^2 Weighting.

Figure 5. Fractional Shift Measurement $\hat{\delta}$ vs. True Fractional Shift δ for Phase Correlation with 3-Point Parabolic Peak Interpolation for Two Different Weightings.

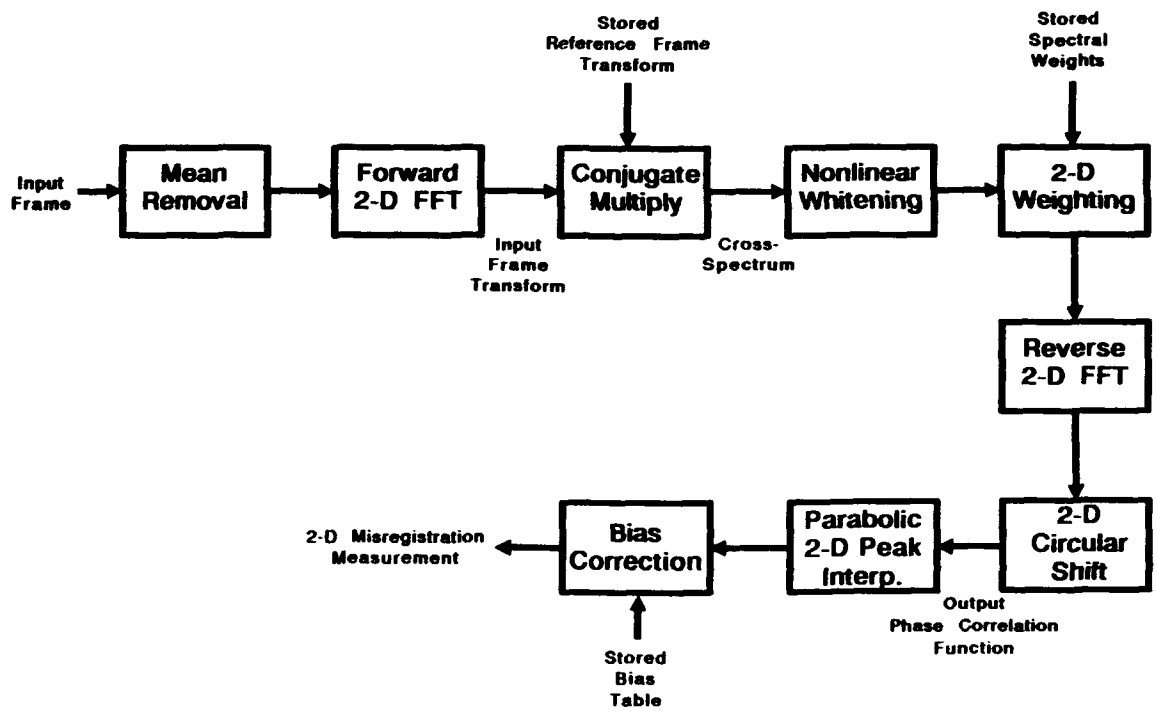


Figure 6. Block Diagram of a Frame-Registration Algorithm Based on the Phase-Correlation Method.

capable of producing nearly perfect sub-pixel jitter estimates for a pair of frames dominated by background clutter. However, in cases where extreme jitter or drift over many pixels may be encountered, the circular wraparound in the periodic correlation may lead to unpredictable measurement results.

The wraparound problem can be avoided by zerofilling each frame to a larger dimension N>M prior to the forward FFT. In particular, if the maximum frame-to-frame drift is limited to ±K pixels in either direction, the value of N should be chosen such that N>M+K. The obvious cost of doing this is the increased computation associated with the larger FFT size. However, there is another, more subtle effect which must also be considered. This has to do with the fact that the discrete Fourier transform of a bandlimited, sampled image with zerofill is equivalent to an interpolation of its periodic components in spatial frequency, and such interpolation cannot be implemented exactly with a frame array of finite spatial extent. Interpolation noise in the phase of the whitened cross-spectrum may lead to unpredictable errors in measuring the location of the correlation peak. Fortunately, it has been found empirically that this problem can at least be mitigated through the use of appropriate spectral weighting.

Although either method can be designed to work well, it would be desirable to use circular correlation whenever possible to minimize real-time computation. Intuitively, one would think that circular processing might be acceptable so long as the frame-to-frame displacements are not a significant fraction of the total frame dimension. This notion has been confirmed by simulation experiments which have been performed.

4.3 Velocity Filtering

4.3.1 Velocity Filter Concept. Figure 1 shows the algorithm chain for a typical application of velocity filtering in an electro-optical sensor system. The output of the focal plane array is assumed to have been pre-processed to provide gain/offset correction, time-delay-and-integration or TDI (in the case of a scanning sensor), and radiation hit suppression. Detection and track initiation functions consisting of frame registration, clutter suppression and velocity filtering are then performed in a high-performance signal processor, after which tracking, discrimination, and other higher-level functions are carried out in a conventional general purpose processor. In this section we are concerned exclusively with the velocity filtering function, which is computationally the most intensive of the operations shown in Figure 1 and which requires a large amount of hardware using existing processor architectures.

The bank of 3-D velocity filters is employed to detect objects moving against the whitened clutter background. For sub-pixel or "point" targets, the hypothesis testing technique used in velocity filtering [4] involves comparisons of integrated pixel intensity along candidate trajectories determined as follows:

a) Shift each frame with respect to its predecessor by a distance equal to an assumed 2-D velocity times the inter-frame period;
b) Add the shifted frame to a running sum frame.

After all frames in the sequence have been processed, the integrated intensity at each pixel in the output frame is compared with a fixed threshold. If the intensity exceeds the threshold, the hypothesis that the corresponding trajectory contains a target is accepted and a target is declared; otherwise, the target

hypothesis is rejected. The foregoing procedure, repeated for each assumed 2-D target velocity, implements the velocity filter "bank" shown in Figure 1.

For additive targets moving against a whitened background, the velocity filtering process provides an effective improvement in the signal-to-noise ratio (SNR) proportional to the square root of the number of frames integrated. Processing an image sequence of length 10 frames, for example, increases the SNR by a factor of $\sqrt{10} \approx 3.2$ or 10dB. The SNR gain leads to improved target detectability relative to that obtained with single frame thresholding.

Velocity filtering also provides the sensor-apparent 2-D position and velocity information needed for automatic track initiation. This feature is inherent to the use of a filter bank which partitions the apparent velocity space: the individual filters tend to separate the targets in accordance with their 2-D velocities. The velocity filter approach is particularly attractive for applications involving multiple targets, since the basic computations required are independent of the number of targets.

Velocity filter computations have the following general characteristics:

a) They utilize simple operations (primarily fetch, add and store) at very high rates;
b) They require very few data-dependent branching operations (until final thresholding);
c) They employ very large data sets.

Because of these characteristics, it turns out that the processor hardware requirements are dominated by memory. One reason for this is the fact that for a fixed memory access time, an increase in the required number of memory accesses per unit time can, beyond a certain point, be accommodated only by an increase in total memory size. For example, in order to meet the processing throughput requirements, the same data may have to be stored redundantly in duplicate memory banks, each of which is connected to a separate processor matched in speed to the memory bandwidth. A second reason is that the total hardware cost measured in terms of number of chips, power consumption, etc., is nearly always dominated by a processor's memory rather by its computational or communications capabilities. For instance, in the SCC-100 parallel processor, developed by Space Computer Corporation specifically for image sequence signal processing, memory accounts for 80% of the total number of chips and 75% of the total power consumption [5].

As a consequence of the above considerations, efforts on velocity filter algorithm design and architecture development must emphasize minimizing both the number of memory accesses and the total memory size.

4.3.2 Velocity Filter-Bank Design. The most obvious way to reduce the amount of hardware needed to implement a velocity filtering scheme is to minimize the number of filters which must be computed. This requires a systematic procedure for the design of filter banks to meet specified velocity coverage requirements.

The basis of the velocity filter approach for detecting moving targets in an image sequence is the matching of hypothesized trajectories to the observed image data. The hypothesized trajectories are assumed to be straight lines originating from the pixels of the initial image (corresponding to constant

target velocity during the observation time). The number of trajectories per pixel, i.e., the trajectory density, must be made large enough so that the performance loss due to mismatch between any actual trajectory and a hypothesized trajectory will be acceptably low. The complete set of candidate trajectories for any pixel constitutes a "velocity filter bank." System design guidelines leading to good performance with a computationally-efficient implementation of the filter bank are:

a) Maintain background clutter near zero apparent velocity by frame registration (by either image shifting or by sensor slewing);
b) If possible, select the frame rate such that the minimum-velocity target moves at least one resolution cell during the inter-frame period;
c) Choose the focal plane integration time so that all targets remain in their resolution cells during this period (to prevent target "streaks" from occurring).

The number of velocity filters required for a particular application (i.e., the number of candidate trajectories per pixel) is determined by the target velocity uncertainty and the velocity filter resolution. For a given filter, the normalized velocity mismatch magnitude $\overline{\Delta v}$ is defined by

$$\overline{\Delta v} = \Delta v \, KT/\sigma_b \tag{10}$$

where Δv is the apparent velocity mismatch magnitude, K is the number of frames filtered, T is the inter-frame period, and σ_b is the one-sigma Gaussian blur circle radius. A plot of peak filter response versus normalized velocity mismatch is shown in Figure 7 with the number of frames filtered as a parameter. These curves are derived on the basis of a Gaussian-shaped target of "width" σ_b and the shift-and-add implementation of the velocity filter. The nominal filter passband (half-power width) can be determined from Figure 7 by locating the point at which the peak filter response is attenuated by a factor of $1/\sqrt{2} = 0.707$. As can be seen from the figure, this occurs at the normalized mismatch $\overline{\Delta v} = 3$, so that from (1) the two-sided half-power width (in velocity units) is given by

$$\Delta v_{3dB} = 6\sigma_b/KT. \tag{11}$$

It is assumed that the sensor is designed so that the optical image is spatially bandlimited by the resolution of the optics, and is sampled by the focal-plane detector array in accordance with the Nyquist criterion. This generally requires a pixel size approximately equal to the blur circle radius σ_b. For such a sensor the above equation shows that the nominal velocity filter passband for K = 10 frames is 0.6 pixels per frame.

The velocity filter bank can be designed by close-packing circles with diameter Δv_{3dB} to completely cover the area representing the range of possible target velocities in 2-D velocity space, as illustrated in Figure 8. System and filter bank design parameters for this example can be summarized as follows:

a) Blur circle radius (σ_b) = 1 pixel;
b) Number of frames (K) = 10;

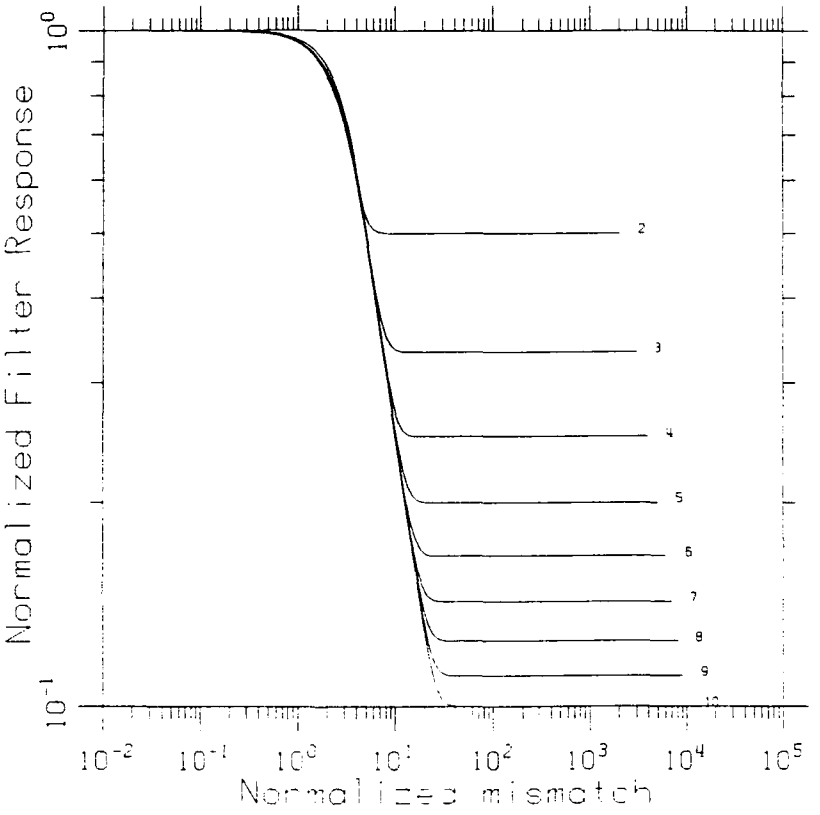


Figure 7. Velocity Filter Response as a Function of Normalized Velocity Mismatch $\overline{\Delta v} = \Delta vKT/\sigma_b$ with Number K of Frames Filtered as a Parameter.

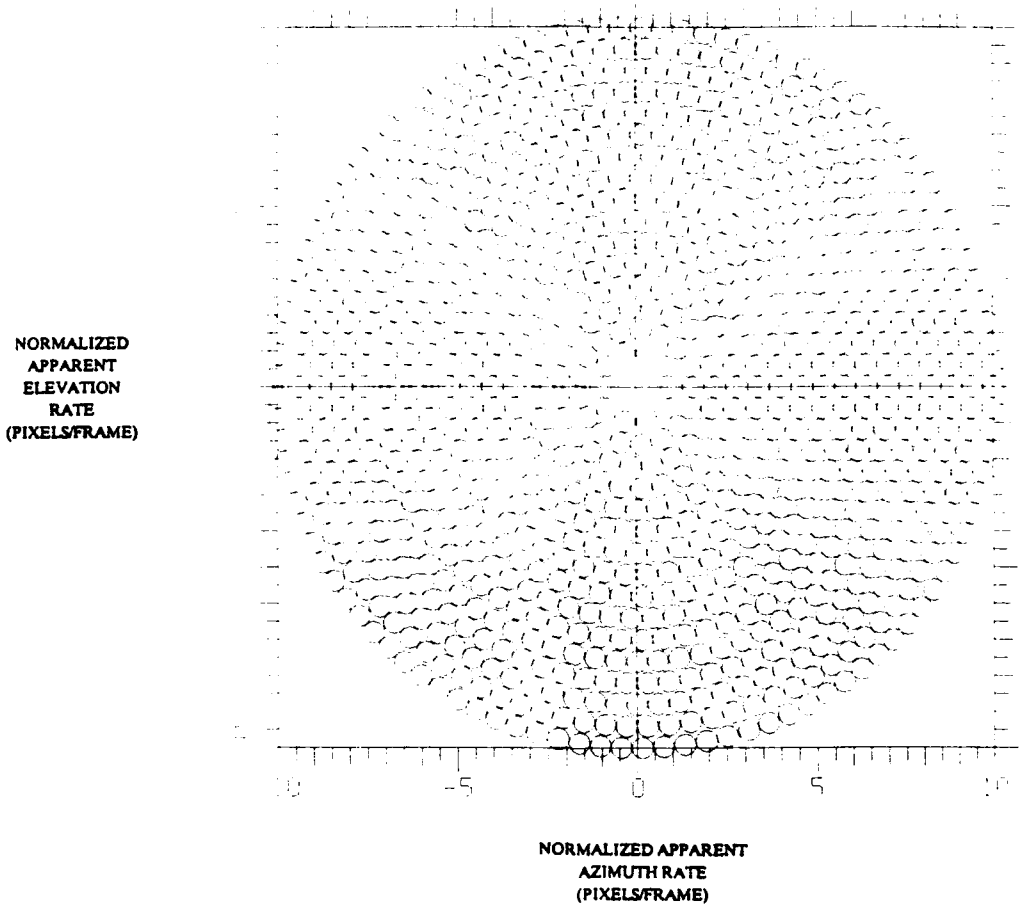


Figure 8. Filter Bank Design for 10-Frame Velocity Filters.

c) Maximum signal-to-noise ratio loss = 0.5 (3 dB);
d) Minimum apparent velocity = 1 pixel per frame;
e) Maximum apparent velocity = 10 pixels per frame.

The total number of filters required for this case is 921.

The use of 3-dB nominal filter spacing is a common practice in the design of filter banks for related applications such as pulse-Doppler radar signal processing. Although the peak signal loss in any one filter is 3 dB, the effective loss in detectability can be considerably less than this. This is because a target with velocity midway between several adjacent filters has a chance of being detected in multiple filters. A detection analysis typically shows that the use of 3-dB filter spacing provides an *average* loss in sensitivity of less than 1 dB relative to the perfectly matched case. For example, to achieve a detection probability P_d of 0.9 at a false alarm probability of 10^{-6}, the SNR at the output of a matched velocity filter must be about 15.5 dB. If this SNR drops to 12.5 dB (i.e., 3 dB down) due to velocity mismatch, the single-filter detection probability drops to 0.32. However, if the same target has an independent chance of being detected in four adjacent filters (at 12.5 dB SNR per filter), then the probability of a detection occurring in at least one of the filters is $1-(1-0.32)^4=0.79$. This corresponds to an effective loss in sensitivity of only 0.7 dB relative to the matched case. Thus, the use of 3-dB nominal velocity filter spacing appears to be a practical compromise between the conflicting desires for low average mismatch loss and a small number of filters.

Figure 9 shows the number of velocity filters required as a function of the number of frames and the radius of the apparent velocity uncertainty region (in pixels per frame) for a sensor with σ_b = 1 pixel. For an apparent velocity range between 1 and 10 pixels per frame, with K = 10 frames, it can be seen that the approximate number of filters required per pixel is between 10 and 900. Since the velocity resolution (11) is inversely proportional to the number of frames, the number of filters grows as the square of the number of frames processed (for a fixed velocity uncertainty region).

4.3.3 Streak Detection. An alternative approach to the "dwell in cell" filter design (in which the focal-plane detector integration time is chosen to be short enough so that a maximum-rate target moves by no more than one detector width during the integration time) is "streak detection," in which the integration time is set to equal to the full inter-frame period, and faster targets streak over multiple detectors during the integration time. Filter design for this case has been studied in [6].

The basic difference between the two approaches is that for the dwell-in-cell approach each velocity filter covers a circular region of the angle-rate uncertainty region, and the filter output increases with the number K of frames processed as $\sqrt{K}$ for all target velocities, while for the streak-detection approach each velocity filter covers a larger oblong region, and the filter output is proportional to $\sqrt{K/v}$ where v is the normalized target velocity in pixels/frame. Thus, the streak-detection approach requires fewer filters, but provides less processing gain for higher-velocity targets. If the higher angular velocities are attributable to the targets being closer to the sensor, and therefore stronger, then the streak-detection approach is the appropriate one.

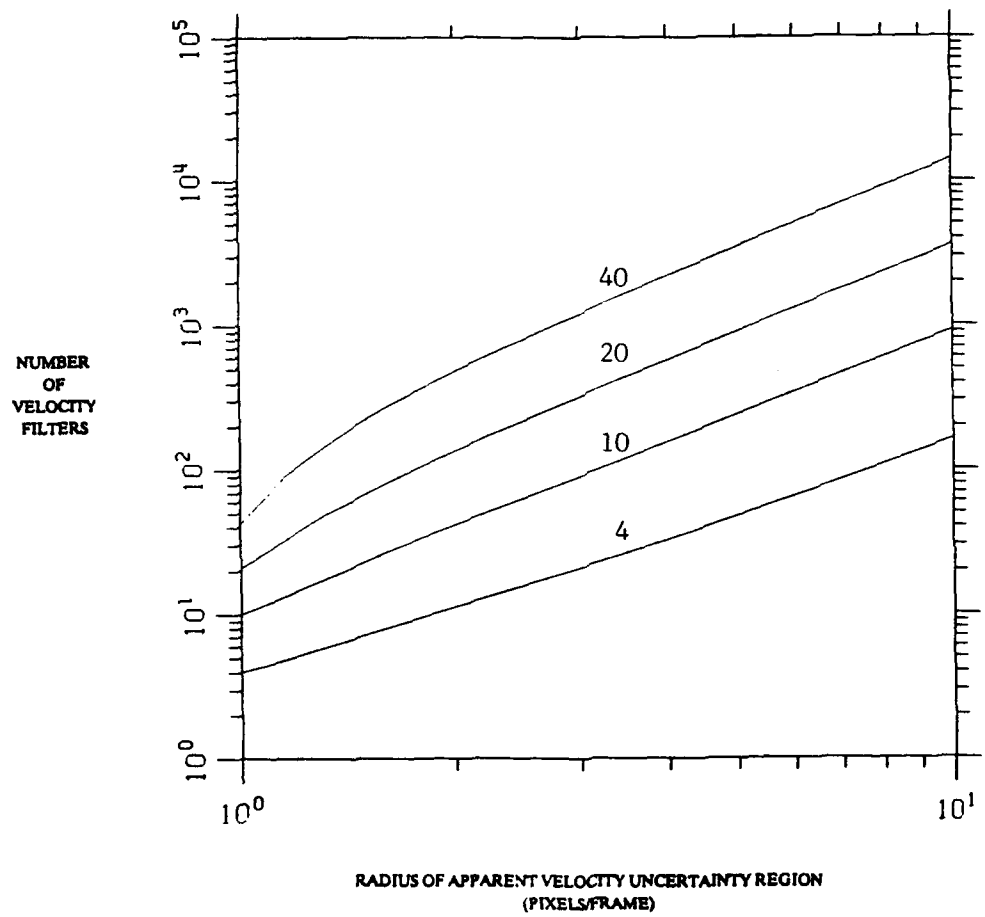


Figure 9. Filter Bank Size vs. Radius of Velocity Uncertainty for K = 4, 10, 20, and 40 Frames (σ_b = 1 Pixel Case).

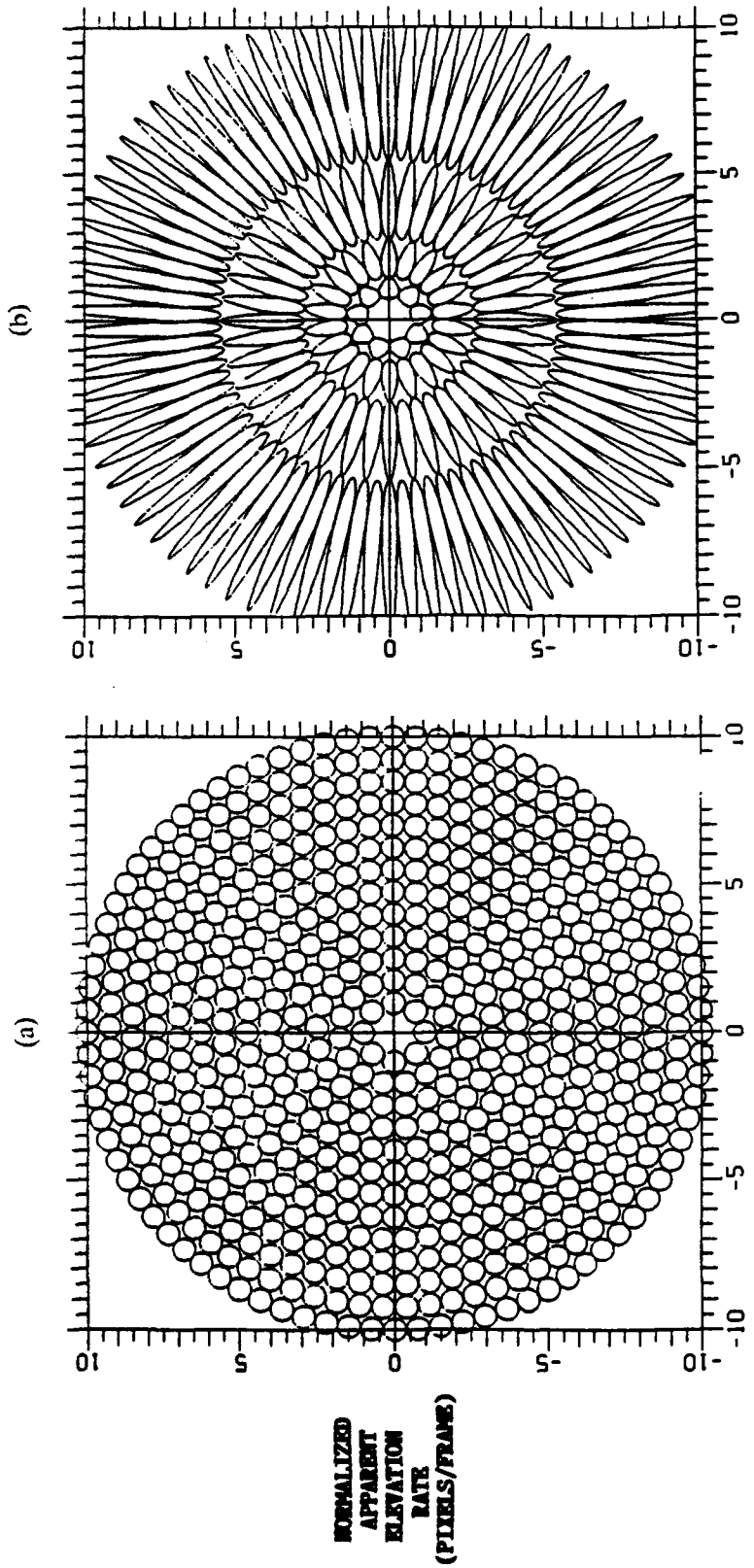


Figure 10. Filter Bank Design for 4-Frame Velocity Filtering for (a) Dwell-in-Cell Sensor, and (b) Streak-Detection Sensor.

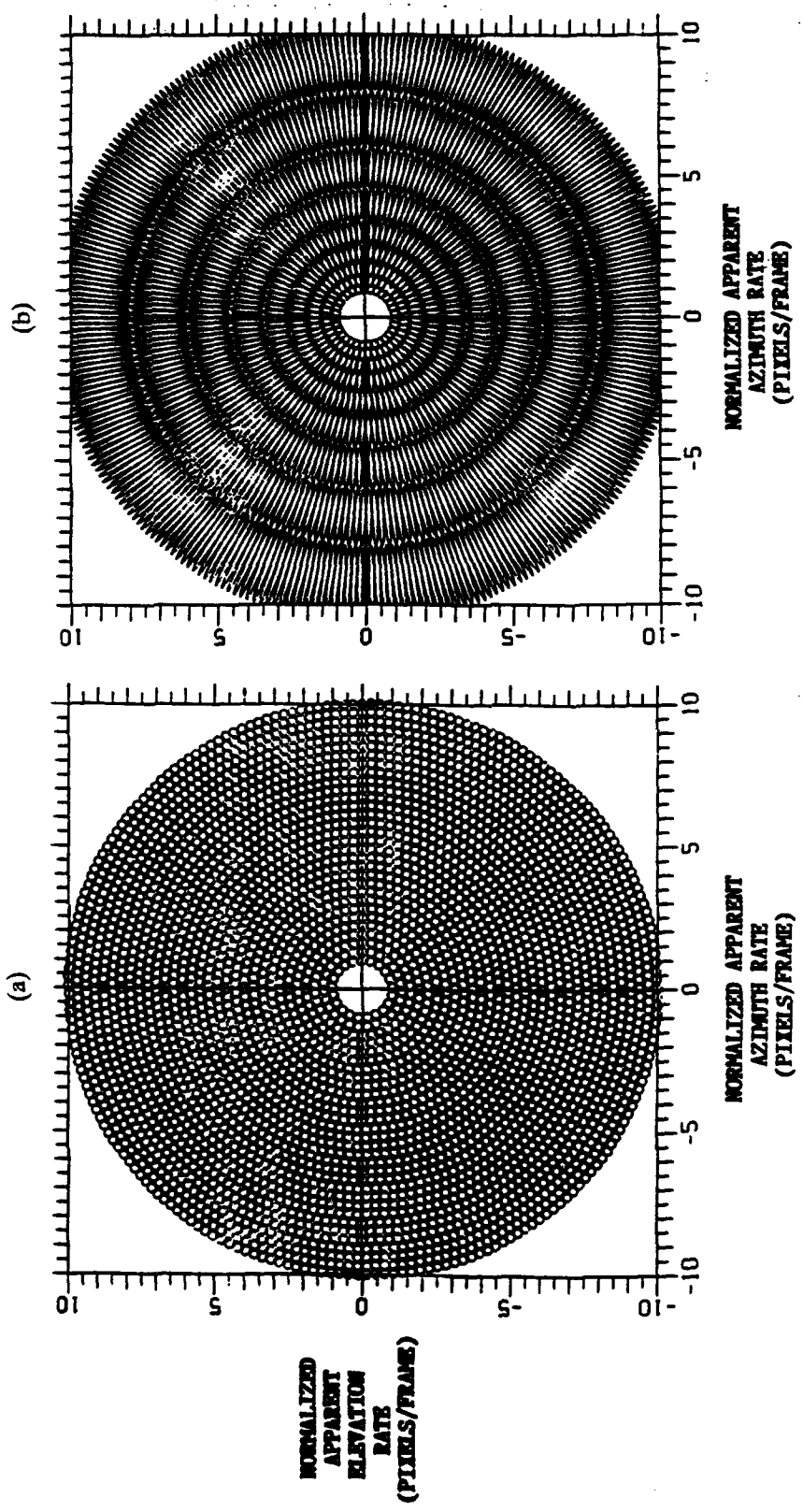


Figure 11. Filter Bank Design for 10-Frame Velocity Filtering for (a) Dwell-in-Cell Sensor, and (b) Streak-Detection Sensor.

developing a real-time moving target detection system has shown that the hardware requirements for velocity filtering are driven primarily by the need for extremely high rates of memory access, particularly when the number of filters is large. In fact, the velocity filter bank is a computational problem that is largely memory access limited rather than compute bound. This characteristic has a major impact on processor cost since memory is an expensive system element in terms of chip count and power consumption.

We have found that the hardware requirements for 3-D velocity filtering can be reduced by several orders of magnitude by carefully managing the size of the filter bank and by devising algorithms, computational strategies and architectures that minimize the required memory size and the number of accesses to memory.

4.4.1 Computational Strategies. The maximum-likelihood hypothesis testing procedure which is the basis of velocity filtering consists of integrating the intensity along candidate trajectories through the frame sequence, and comparing the result to a fixed detection threshold. This is equivalent to testing for a nonzero mean intensity in a sequence of i.i.d. Gaussian pixel observations along each assumed path through the frame sequence (or "stack"). A basic implementation issue is the strategy for carrying out the pixel integration process for the multiple trajectories that constitute the velocity filter bank. Figure 12 illustrates three distinct alternatives that can be considered. In the first approach shown in Figure 12(a), the states (or partial sums) for each trial trajectory at every pixel in the current frame are retained in memory. This is referred to as the *frame-first* strategy, since all of the velocity filter states at each pixel in a frame are updated before going on to the next frame in the sequence. Recursive velocity filtering algorithms are a prime example of the frame-first computational approach. Figure 12(b) shows a *pixel-first* strategy, in which the filter trajectories starting at a given pixel in the first frame are traced all the way to the bottom of the frame stack and thresholded before moving on to the next pixel. Outputs for the "tree" of trial paths that constitute the entire filter bank are thus computed on a pixel by pixel basis. The third approach, referred to as the *filter-first* strategy and sketched in Figure 12(c), computes path sums along trajectories of the same slope (or 2-D velocity) emanating from every pixel of the initial frame. Multiple filters in a bank are computed back-to-back by repeating the same procedure using different sequences of 2-D frame shifts. The filter-first approach leads to the intuitively-appealing "frame shift-and-add" interpretation of velocity filtering. It is also readily implemented on array processors, since each filter can be computed by applying a sequence of vector shift-accumulate operations to ordered pixel data stored in a frame buffer.

Although all three methods can be designed to yield the same results, there are major differences among them from an implementation standpoint. For example, in frame-first processing, the intermediate states of all velocity filters at every pixel must be stored, whereas with pixel-first or filter-first processing it is necessary to store the input frame stack. In many surveillance applications there are hundreds or thousands of velocity filters but only a few tens of frames being integrated; thus it is clear that the amount of memory is minimized by storing the frames rather than the filters. For this reason alone, the use of filter-first or pixel-first computational strategies will generally lead to more economical hardware implementations of large-scale filter banks.

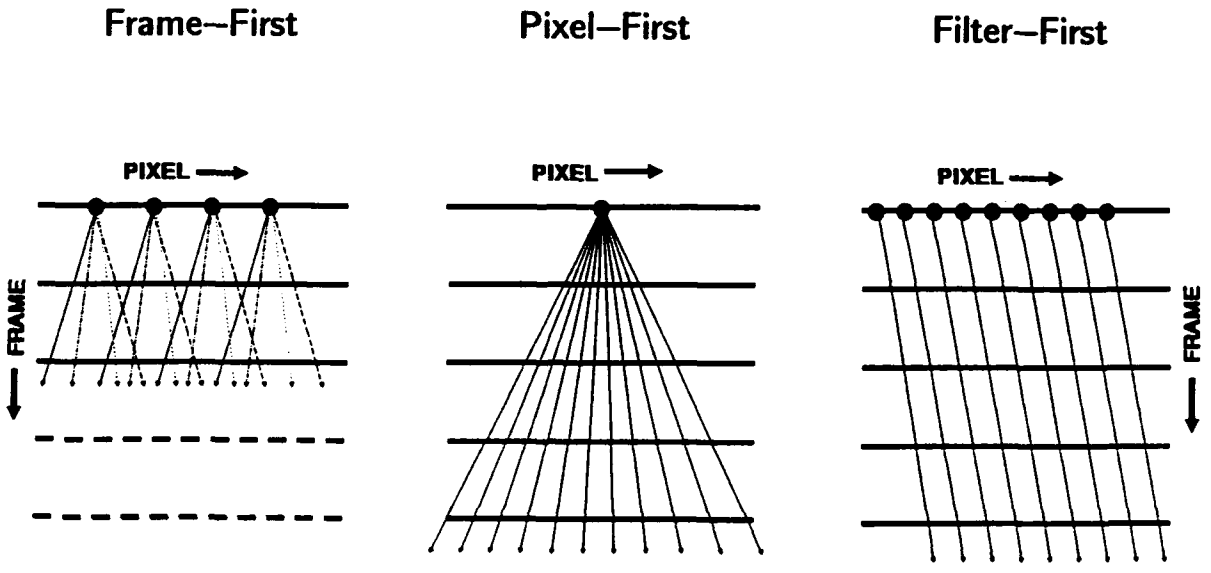


Figure 12. Filter Computation Strategies.

The only disadvantage of the filter-first and pixel-first strategies is that filter computation cannot begin until the entire frame stack has been collected and stored. In contrast, the frame-first approach allows the input images to be processed on the fly as they are received. This makes the frame-first strategy attractive for applications where processing latency is critical and the number of filters is relatively small (e.g., closed-loop single-target tracking). However, in long-range surveillance systems that employ large filter banks, one would probably be willing to trade a reasonable amount of processing delay in exchange for a many-fold reduction in processor memory size.

4.4.2 Alternative Algorithms. The velocity filtering concept discussed so far involves space-time domain integration of word-precision pixel intensities along hypothesized trajectories through a frame stack. Two important variations of this basic algorithm were examined in detail because of their potential for simplifying the filter computation and reducing memory access and data storage requirements with minimal performance loss. These alternative algorithms, the sequential velocity filter and the single-bit velocity filter, are discussed in turn below.

Sequential Velocity Filter. The basic velocity filter is a "fixed sample size" hypothesis testing procedure in which the integrated intensity along each trajectory is compared with a single detection threshold after all frames in the sequence have been processed. Since this requires at least one memory access per pixel per frame per velocity filter, the total number of memory accesses per second can become very large for typical acquisition scenarios. A modification of the basic velocity filter which reduces the number of memory accesses and computations has been developed by S.D. Blostein and T.S. Huang [11]. The key feature of this approach is the use of a truncated sequential hypothesis test in place of the conventional fixed sample size test. Truncated sequential testing is a recently developed variation of Wald's classical Sequential Probability Ratio Test (SPRT) [12] in which it is assumed that the test is terminated after a specified number of stages. The truncated SPRT was originally defined by Tantanatara and Poor [13] in terms of a "mixture" between a standard fixed sample size hypothesis test and a pure sequential test.

Figure 13 illustrates a truncated sequential testing approach in which the cumulative intensity along a given trajectory is compared with two thresholds after each new frame pixel is added to the running sum. If the integrated intensity exceeds the upper threshold, a target is immediately declared. If the integrated intensity falls below the lower threshold, the hypothesis that the trajectory contains a target is rejected, ending the test for that pixel and that velocity. Otherwise, the intensity value falls between the two thresholds and the decision is deferred: the test continues to a later stage and the trajectory state is stored. If the test reaches the last frame a forced decision is made between the two hypotheses, i.e., the test sequence is truncated. This procedure permits decisions to be made earlier than the last frame, so that on average it is necessary to process fewer pixels per trajectory (although it should be noted that the maximum number of frames in a truncated sequential test must be slightly larger than the length of the equivalent fixed size test).

Using results given in [13], expressions can be derived for the maximum test length or number of SPRT stages (K) and the upper and lower thresholds for detecting an additive point target in white Gaussian noise. The specified SPRT design parameters are the single-frame target SNR and the desired probabilities of false alarm (α) and detection (β). The relevant performance parameters for the truncated SPRT are as follows:

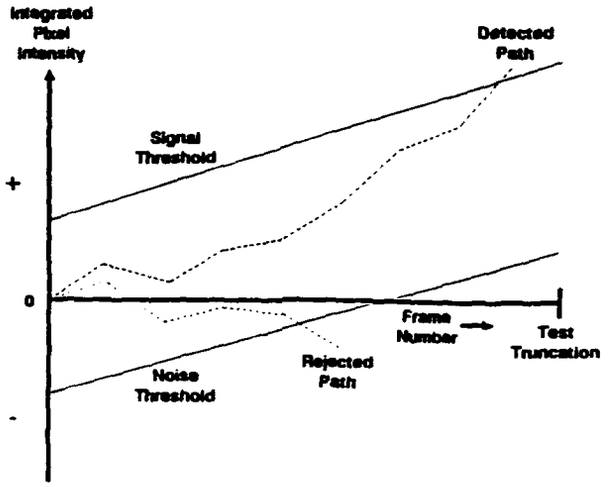


Figure 13. Sequential Hypothesis Testing Concept.

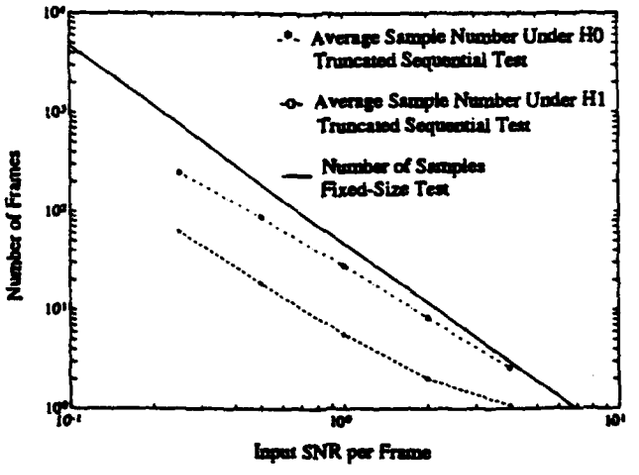


Figure 14. Comparison of Fixed Sample-Size Test and Truncated Sequential Test for $P_d = 0.9$ and $P_{fa} = 10^{-8}$.

ASN_0 = the average sample number under the noise-only hypotheses H_0 (the expected number of stages required to terminate the test under H_0);

ASN_1 = the average sample number under the target-plus-noise hypothesis H_1 (the expected number of stages required to terminate the test under H_1);

P_{fa} = the actual false alarm probability of the test;

P_d = the actual probability of detection of the test.

The average sample numbers ASN_0 and ASN_1 are calculated in terms of the discrete probabilities that the sequential test reaches a given stage or frame k (k=1,...,K) under the hypotheses H_o and H_1, respectively. These so-called stage probabilities can be evaluated analytically since the sequence of partial sums computed along any trajectory in a stack of whitened frames is a correlated Gaussian random process. The actual detection performance of a truncated SPRT (P_{fa} and P_d) will differ from the specified design performance (α and β), although the design is theoretically conservative in the sense that $P_{fa} \leq \alpha$ and $P_d \geq \beta$. However, the exact performance of a test can be predicted by evaluating the Gaussian conditional probabilities that each hypothesis is accepted, given that the test reaches a particular stage.

Figure 14 shows a direct comparison of two test procedures for an *actual* probability of detection of 0.9 and a probability of false alarm of 10^{-8}. The plots show the number of frames required to achieve this detection performance as a function of the input amplitude signal-to-noise ratio (SNR) per frame. In the case of the fixed sample-size test (solid line), the ordinate corresponds to the total number of frames integrated. In the case of the truncated SPRT (dashed lines), the ordinate corresponds to the *average* number of frames which must be processed. Two curves are shown for this test. The upper one (H_1) indicates the average number of frames required to declare a target when one is present (ASN_1); this value is slightly below that required for the fixed sample-size test. The lower curve (H_0) indicates the average number of frames required to reject the target-present hypothesis (ASN_0); this number is considerably below that for H_1.

For illustration, assume that the input SNR is equal to 2 (6 dB). Figure 14 shows that a fixed sample-size test requires 12 frames to achieve the assumed detection performance, while the truncated sequential test (H_0 curve) requires slightly more than 2 frames on the average. Averaged over a large number of trial trajectories, most all of which contain only noise, the number of memory accesses required for the truncated sequential test will be about one-sixth the number required for the equivalent fixed sample-size test. Simulations of sequential velocity filtering on synthetic and real image sequences of length 10 to 50 frames have confirmed predicted reductions in the number of pixel computations by factors of 4 to 7 relative to fixed size tests of equivalent performance. These reductions are the main motivation for considering a sequential velocity filtering approach.

Sequential velocity filtering is most effectively implemented with the pixel-first strategy illustrated in Figure 12(b). This makes it possible to exploit cases where adjacent trajectories in the filter bank share the same quantized path for the first several frames before diverging. If the hypothesis test for a given trajectory is terminated early, a linked list can be used to quickly eliminate other filters which share the same trajectory segment. However, for small to moderate numbers of frames and a properly designed filter bank, opportunities for such "tree pruning" will probably occur only on the first or second frames.

Single-Bit Velocity Filtering. As previously noted, in conventional processor architectures the amount of hardware required for velocity filtering is dominated by memory. It is therefore natural to consider the possibility of using reduced-precision image data. This can be accomplished by quantizing the whitened images to only a few levels, and in the limit, to only two levels (i.e., by using single-bit data). This technique was in fact employed in the binary moving-window detector (also called the M-out-of-N or coincidence detector) first used in radar systems many years ago when digital processing technology was relatively primitive [14]. More recently, Chu [15] and Preston [16] have considered the use of one-bit data for moving object detection.

Figure 15 shows a diagram of the single-bit filter concept. The pixel values along a particular trajectory through a stack of K whitened frames are denoted as x_k, k=1,...,K. Prior to filtering, these data are hard-limited by a one-bit quantizer with normalized threshold t, which produces binary-valued pixels z_k according to the rule

$$z_k = \begin{cases} 1 \; ; x_k > t\sigma_k, \\ 0 \; ; x_k \le t\sigma_k, \end{cases}$$

where σ_k is the rms level of the zero-mean noise background in the pixel containing x_k. The single-bit velocity filter is then equivalent to a binary integrator which sums the K bits along a trajectory to produce an output count Z, which is compared with a count threshold k_0 ($1 \le k_0 \le K$) for detection.

Two thresholds, a *quantization threshold* t and a *count threshold* k_0, are chosen to design a single-bit velocity filter. A reasonable design criterion is to optimize filter sensitivity at a specified level of performance. This leads to the following design procedure: for a specified detection performance (P_d,P_{fa}) and number of frames integrated (K), minimize the required SNR through proper selection of the thresholds t and k_0. This approach also minimizes the number of frames required to achieve the specified performance at a fixed input SNR.

The performance of the single-bit velocity filter for a nonfluctuating point target in white Gaussian noise is found by a straightforward application of Bernoulli probability analysis. Figure 16 shows some results for the case where the performance is fixed at P_d=0.9 and $P_{fa}=10^{-6}$. The curves show the minimum target SNR (in dB) required to achieve this performance as a function of the count threshold k_0 for various numbers of frames K. For example, the points on the curve for K=8 show how the minimum detectable SNR changes as the count threshold k_0 is varied from 1 to 8, with the quantization threshold t adjusted to maintain the desired false alarm probability. Evidently, the detection sensitivity of the 8-frame binary filter is highest when k_0=5 (i.e., a 5-out-of-8 coincidence detection criterion). In fact, each curve shown in Figure 16 has an optimum k_0 which occurs about midway between the extremes of low coincidence (e.g., k_0=1) and total coincidence (k_0=K). The behavior represented by Figure 16 appears to be broadly applicable in cases where high detection performance (e.g., $P_d \ge 0.9$, $P_{fa} \le 10^{-6}$) is required. Thus, as a general rule for single-bit velocity filtering of K frames, one should choose the count threshold to be k_0=K/2+1; then calculate the binary quantization threshold t that achieves the desired false alarm probability.

The use of one-bit frame data in the velocity filter leads to a loss in detection performance relative to a full-precision implementation. Figure 17 shows a direct performance comparison for the case where

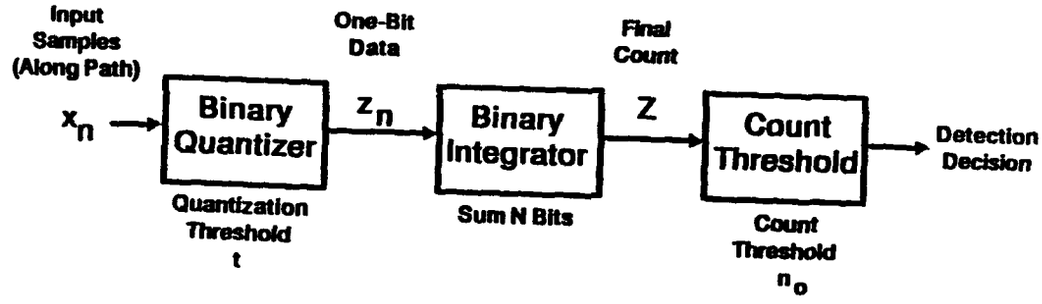


Figure 15. Single-Bit Velocity Filter Concept.

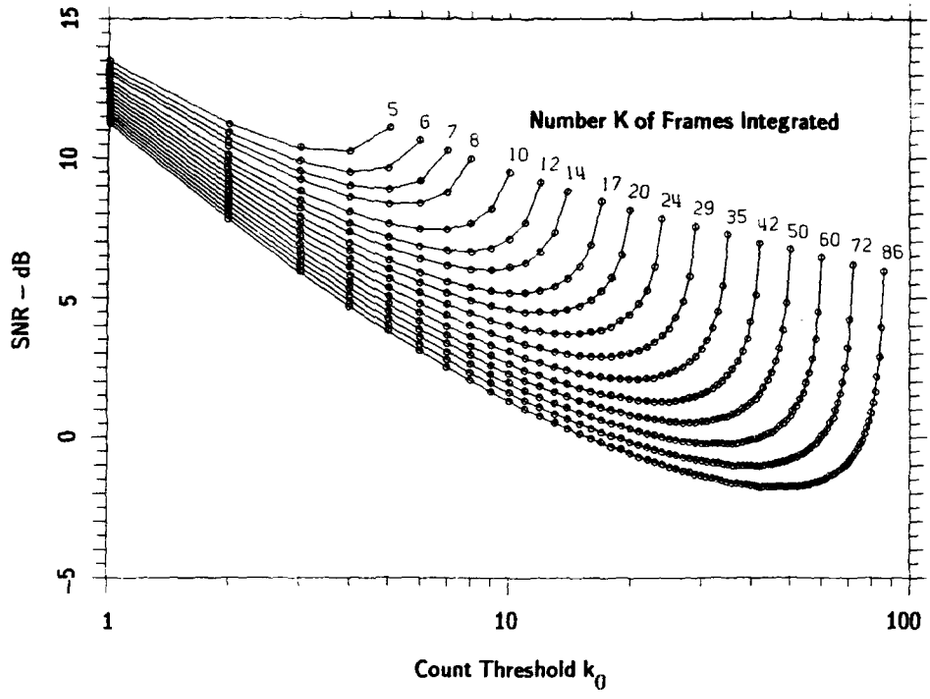


Figure 16. One-Bit Filter Design Optimization for $P_d = 0.9$ and $P_{fa} = 10^{-6}$.

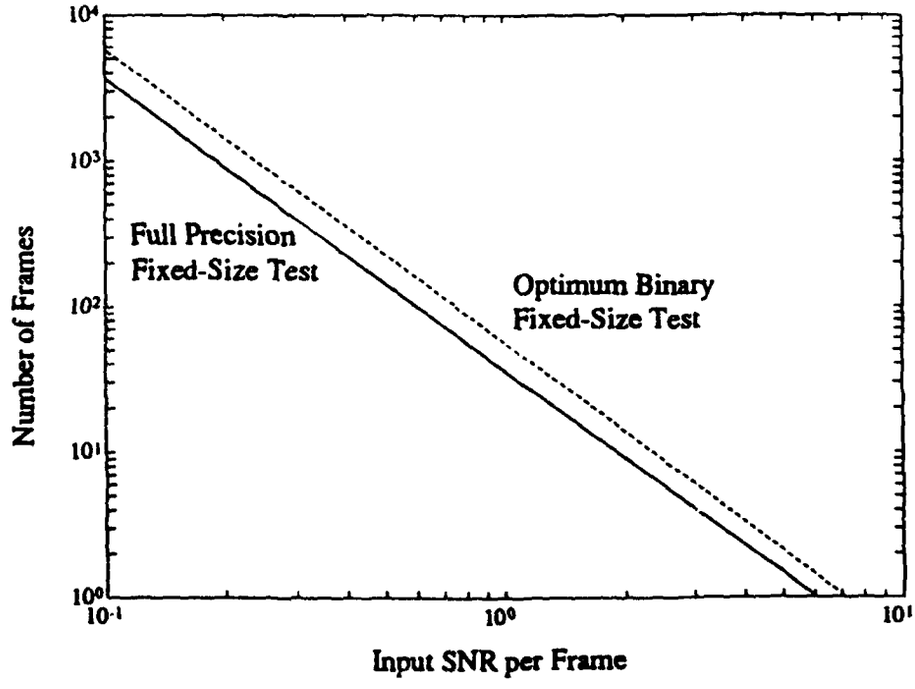


Figure 17. Velocity Filter Performance Comparison for $P_d = 0.9$ and $P_{fa} = 10^{-6}$.

P_d=0.9 and $P_{fa}=10^{-6}$. The solid line gives the number of full-precision frames required to achieve this performance vs. the target input SNR, while the dashed line above shows the same relationship for the optimum single-bit velocity filter. For a fixed number of frames K, the effective SNR loss due to optimum binary quantization is a factor of about 1.26, or 2 dB. The quantization loss would lead to a degradation in detection performance if nothing were done to compensate for it. In order to obtain the *same* performance as a full-precision velocity filtering system, it is necessary to integrate enough additional frames to obtain a further processing gain of 2 dB (a factor of 1.26). This requires an increase in the number of frames processed by a factor of $(1.26)^2=1.58$. Moreover, since the size of the filter bank grows as the square of the number of frames (for a fixed velocity coverage), the total number of filters must be increased by a factor of $(1.58)^2=2.5$. Thus, the total number of "operations" required to implement a single-bit filter bank is $(1.58)^3\approx4$ times that of a full-precision system with equivalent performance. However, since these are *one-bit* operations, major reductions in the amount of computational hardware and memory can be realized even after accounting for the increased numbers of frames and filters.

Algorithm Comparison and Selection. Through proper design it is possible to achieve essentially the same detection performance with the basic, sequential or single-bit velocity filtering approaches. Therefore, the choice of which algorithm to use in real-time applications can be governed by hardware considerations; primarily, the suitability of the various methods for implementation in high-speed special-purpose processors dedicated to the velocity filtering function.

On this basis, the sequential algorithm was found to have a key disadvantage, even though it reduces the average number of computations per filter by an factor of 5 or more relative to the other methods. The drawback is that the SPRT is inherently a data-dependent procedure which requires an unpredictable pattern of access to the pixel data in the frame stack. In a single-processor system (e.g., a minicomputer), this is not a problem since only one pixel can be randomly accessed and processed at a time. However, the requirement for data-dependent processing along every filter trajectory makes it difficult to fully exploit the concurrency available in special-purpose processors that can rapidly apply fixed sequences of operations to vectors of pixels from the frame stack. From this viewpoint, sequential algorithms appear to provide substantially less payoff than fixed sample size algorithms for real-time applications. Nonetheless, it should be noted that sequential testing provides perhaps the best approach for implementing large numbers of velocity filters in a general-purpose computer.

The selection of full-precision vs. single-bit data in a fixed size test was examined in detail, and led to the choice of a filter-first single-bit approach as the best algorithm for real-time implementation. In terms of data memory reduction, the use of one-bit images for velocity filtering is clearly attractive. In addition, single-bit processing was found to have important architectural advantages which, in terms of hardware, are even more significant than the obvious reduction in the amount of memory required for frame storage. These advantages of single-bit processing hold even though the total number of pixel operations must be increased by a factor of about 4 to obtain the same performance as the basic full-precision velocity filter.

4.4.3 Velocity Filter Processor Architecture. Space Computer Corporation has developed a new special-purpose architecture for single-bit velocity filtering called the *Velocity Filter Processor* (VFP). The

design is based on a concurrent bit-serial processing approach which maximizes the filter computation rate for the following nominal design parameters:

a) Frame size = 256 x 256 pixels (one-bit per pixel);
b) Frame stack size = 16 frames.

The binary frame stack specification of 16 corresponds to an SNR gain of 3.2 (10 dB), which is felt to be a minimum operationally significant "quantum" of velocity filtering.

Concurrent Bit-Serial Approach. Figure 18 shows a basic block diagram of the bit-serial approach to velocity filter bank computation. The single-bit image sequence data are broadcast from a single memory on a bit-serial, pixel-by-pixel basis for all frames *in parallel* to a bank of special-purpose VFP nodes. A key feature of this arrangement is the use of the memory as an active processing element, rather than as a passive storage medium for the input frame sequence. This permits a major savings in hardware, since it is not necessary to provide duplicate frame storage in all of the nodes.

In the filter-first bit-serial approach, 16 bits representing the time-varying binary values for a single pixel in the frame stack are available at the VFP node inputs on each processing cycle. Each node processes these inputs *concurrently* to achieve the highest possible filter computation rate. This approach requires a mechanism for aligning the incoming pixel data in the VFPs to implement the frame shifts required by the velocity filters.

In the concurrent bit-serial processing scheme, each frame is envisioned as a one-dimensional bit stream with pixels arranged in lexicographical order. To implement the filter for a particular trajectory through the frame stack, the input bit streams corresponding to the frames are delayed (or shifted) with respect to one another in accordance with the filter velocity, as shown in Figure 19. Then the bits along each vertical column through the skewed frame stack are added and compared against a count threshold. The output from each node is a set of threshold crossing reports, each consisting of target position index, velocity filter index and count magnitude.

Operational Description. Figure 20 shows a functional block diagram of the proposed concurrent VFP architecture with 16-frame bit-serial input as described above. The VFP has four basic functional elements: a sequencer/controller, a group of delay lines (16 total), a bank of decoders, and a FIFO.

The sequencer controls three distinct operations: write into the delay lines, read out of the delay lines, and add/compare. Each delay line shown in Figure 20 is a memory whose size in bits depends on the maximum frame shift required for the corresponding frame. One frame in the stack (denoted as bit 16 in the figure) is always designated as the reference and need never be shifted; it requires no delay at all in principle. Bit 1 represents the frame which must be shifted the most; it requires the longest delay line. For example, if the maximum frame shift could be as large as half the full frame in either dimension, then the length of delay line 1 would be approximately equal to half the total number of pixels in the frame. For the remaining input bits, the delay line length varies linearly between these two extremes as illustrated in Figure 20.

If the 16 input bits are presented to the VFP in natural frame order, then the delay line system shown in Figure 20 can shift the incoming bit streams in only one direction (i.e., backward in time). To cover 2-D velocities in all possible directions, the VFP must have the capability to shift the frames either

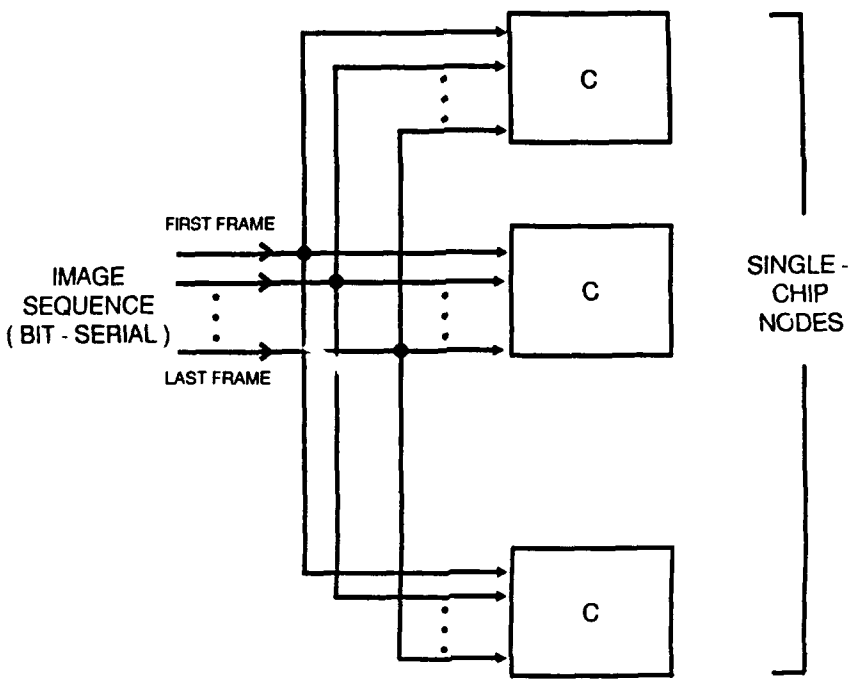


Figure 18. Special-Purpose Bit-Serial Implementation of Velocity Filter Bank.

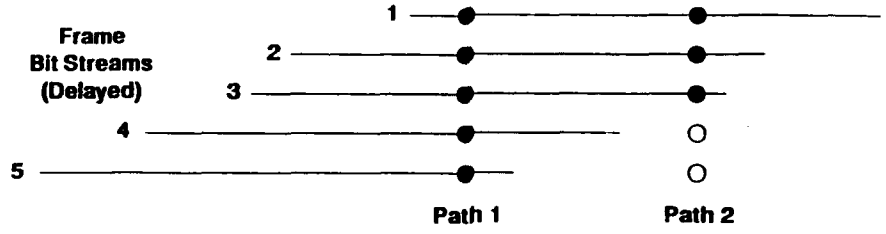


Figure 19. Concurrent Bit-Serial Processing Scheme.

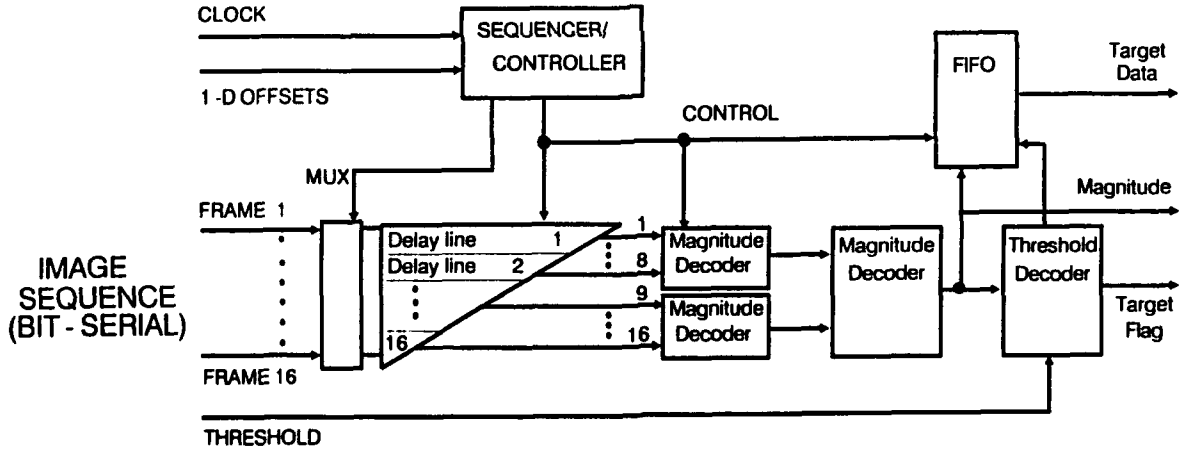


Figure 20. Node Architecture for Bit-Serial Velocity Filter Processor (VFP).

forward or backward. A bit multiplexer (MUX) is therefore included to reverse the order of the input bits when necessary to implement the actual frame shifts required by the filter being implemented.

The sequencer maintains two pointers (read and write) and a counter for each of the 16 delay lines. The write pointer (WP) points to the position on the line into which an incoming bit will be written. The counter is initially set to the number of pixels by which the corresponding frame is to be delayed. After each input bit is processed, the write pointer is incremented and the counter is decremented. The read pointer (RP) initially points to a null position at the start of the delay line and remains there until the counter reaches zero. Thereafter, the read pointer is incremented along with the write pointer after each incoming bit to maintain the appropriate frame delay. Zeros are read from the line while the read pointer is in its initial position (i.e., before the counter decrements to zero).

Reading the 16 delay lines simultaneously transfers 16 bits along a particular trajectory to a three stage table-lookup decoder, which effectively adds the bits and compares the resulting count against a threshold. The first decoder stage is implemented with a pair of 256 x 4 bit RAMs which perform count magnitude table-lookups on bits 1-8 and 9-16, respectively. The resulting pair of 4-bit counts are then combined to form an 8-bit address into the second magnitude decoder (another 256 x 4 bit RAM), which outputs the total count for all 16 bits. The final stage is the threshold decoder, a 16 x 1 bit RAM containing ones and zeros to indicate "hit" or "miss" as a function of the bit count represented by the input 4-bit address. When the threshold decoder produces a "one," indicating that the sum of the bits exceeds the selected count threshold, then the total count (target amplitude) and the current location of the read pointer (target position) are stored in the FIFO as output detections.

This basic cycle is repeated under the control of the sequencer until all the pixels in the frame have been processed. Additional velocity filters are implemented on the stored binary frame stack by resetting the delay line pointers and counter for the new 2-D velocity, and then passing the bit-serialized pixel stream through the device again. Since the the filter set-up time is negligible compared to the filter processing time, it is anticipated that multiple filters in a bank can be computed essentially back-to-back by a single VFP node.

Throughput and Memory. A key feature of the proposed VFP design is its capability to achieve very high filter computation rates on a single-chip device. The filter processing rate is determined by the time required to complete a cycle consisting of a 16-bit delay line write, a 16-bit delay line read, and two 8-bit RAM table-lookups. Ignoring the strong possibility of partial concurrency in these operations, the basic I/O cycle time for the VFP is estimated to be 50 nsec if implemented on a single VLSI chip. The total time required to compute one velocity filter is therefore bounded by (256 x 256 pixels) x 50 nsec = 3.3 msec. This corresponds to a sustained computation rate of at least 300 velocity filters or 20 million target path tests per second on a single-chip node!

The amount of memory for the VFP is dominated by the length of the delay lines, which is in turn established by the maximum frame shift to be implemented. For the proposed design, the maximum shift is specified as 128 pixels in either frame dimension, or one-half of the 256-pixel frame width. This corresponds to a maximum delay of 32K bits. The triangular delay line scheme shown in Figure 20 therefore requires a memory size of approximately 0.5 x 16 x 32K bits = 256K bits, which is close to the current upper limit for on-chip RAM in standard-cell ASICs. The corresponding limit on apparent target velocity is

obtained by noting that a total translation of 128 pixels over 16 consecutive frames implies a maximum target speed of 128/(16-1) = 8.5 pixels/frame in any direction.

Growth Options. The VFP is nominally designed to perform 16-frame single-bit velocity filtering on 256 x 256 input frames. However, it also constitutes a basic building block for systems which employ larger numbers of frames for additional integration gain, and it is fully capable of processing frames larger than 256 x 256 pixels. For example, two VFPs operating in parallel can implement a bank of 32-frame velocity filters. Each VFP performs the frame delay and bit counting functions synchronously for 16 of the 32 frames in the input stack. The output 4-bit counts from the two devices are subsequently processed by an external threshold decoder (a 256 x 1 bit RAM) to generate velocity filter detections. Doubling the number of frames integrated results in a corresponding reduction in normalized velocity coverage, due to the fixed size of the one-bit delay lines in the VFP (16 kbits/line). For the case of 32-frame processing considered here, the maximum target speed would be reduced to 4.1 pixels/frame, about half of the 8.5 pixels/frame coverage obtained for the 16-frame case.

Larger frame sizes can be accommodated in a straightforward manner by simply increasing the size of the input memory which feeds the bit-serialized pixel data to the bank of VFPs. The consequences of an increase in the frame size are reductions in both the filter computation rate and target velocity coverage. For the case of 512 x 512 frames, it is estimated that a single VFP would be capable of computing 75 velocity filters per second. The delay lines would now support a maximum frame shift of 64 pixels in either direction, corresponding to target speed coverage of 4.3 pixels/frame for 16-frame integration.

4.4.4 Velocity Filter Processor Implementation. Although the VFP can in principle be implemented in conventional components, in a gate array, in standard cell or in full custom ASIC design, the standard cell approach appears to be most appropriate given the availability of the required building blocks from chip vendors. For example, LSI Logic's LCB007 series of Cell-Based ASICs, based on a 1-micron HCMOS process technology, appears to be well matched to the VFP requirements. Preliminary chip sizing studies show that the baseline VFP floorplan, including 256K of static RAM, 10K of logic, and all routing and I/O circuitry, will fit inside an area of dimensions 15mm by 15mm, which is within typical design guidelines for standard-cell ASIC technology. For demonstration and ground based test purposes, the device would be packaged in a 2 $in.^2$ ceramic pin-grid array. For future applications with stringent size and weight limitations, one or more VFP chips could be packaged together with interface and support chips in a single multichip module using silicon-on-silicon hybrid wafer-scale integration (HWSI).

4.5 Examples

Algorithm simulation experiments have been conducted on real and simulated passive sensor imagery to validate some of the basic aspects of velocity filter detection, track initiation and discrimination performance. A variety of target and background image scenes have been processed, including a subset of the 1995 Centerline MEA-1 Threat, a 112-target midcourse dataset used for scan-to-scan correlation studies at the MIT Lincoln Laboratory, scenes containing simulated kinetic kill debris, and several sets of digital imagery recorded by the Lincoln Laboratory 30-inch "Quad-Camera" optical telescope in New

Mexico. The latter source of imagery was of particular interest since it was collected on an actual CCD focal plane array. The Quad-Camera datasets which were processed included a sequence of scenes of a fast-moving satellite; these were used for testing the basic velocity filter. Scenes of the Great Nebula in the constellation Orion were also used to study background suppression. All of these scenes exhibited the effects of CCD photodetector noise in addition to a number of other artifacts including image jitter, clipping due to limited dynamic range, and "holes" caused by dead detector cells. Some of the experiments which used the various MIT image datasets are described in the next several sections.

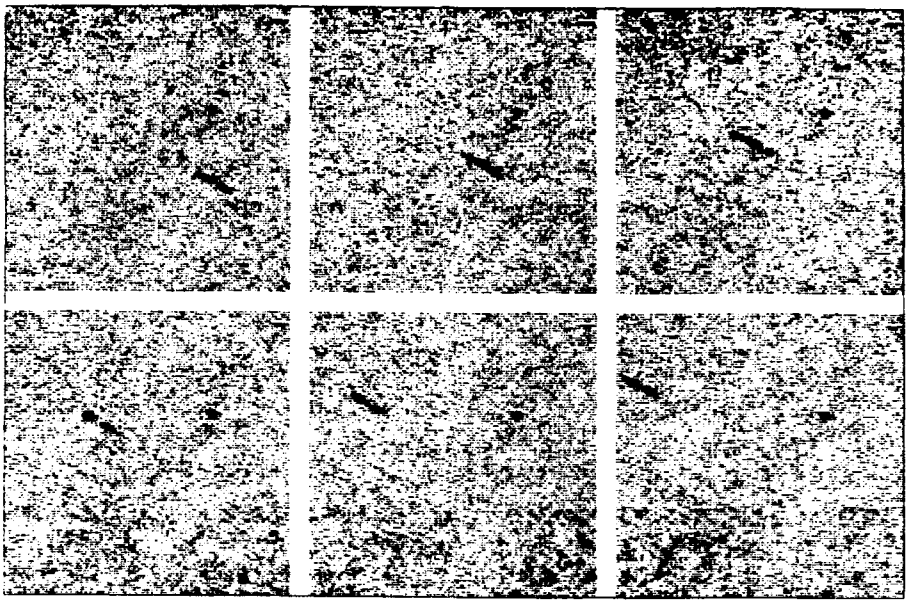
4.5.1 Velocity Filter Detection of a Satellite Streak. The application of velocity filtering to detection of moving objects in passive sensor imagery was investigated by processing a sequence of six staring telescope scenes which are shown in 16-level gray scale format in Figure 21a. An orbiting rocket case appears as a streak in each of the images due to the finite frame exposure time. The brightness fluctuations are caused by tumbling of the rocket case, as shown more clearly in the composite image of Figure 21b. Note also the presence of a very bright fixed star and several vertical bands caused by the CCD readout process.

Prior to velocity filtering, we used 5 additional independent images of the same scene background (taken after the target had moved out of the field-of-view) to degrade the average target SNR to about 3 dB in each frame. These low-SNR images are shown in Figure 22. Figure 23 shows the step-by-step results of processing on input frame number 6 in Figure 22. Figure 23b is the same image after removal of the nonstationary median in each pixel, which was estimated from the sequence of 6 images as a surrogate for the pixel mean. The median filtering operation suppressed the fixed star clutter and other spatially-varying background artifacts, and created a homogeneous background. Figure 23c shows the composite frame that resulted from shifting-and-adding the six frames of Figure 22 according to the known target velocity. Note that the target is just visible in that frame (compare to last frame of Figure 21a). Correlating the data of Figure 23c with a streak template representing the average target signal on a single frame produced the final output shown in Figure 23d. Comparison of Figure 23d with the original frame 6 image in Figure 21a shows that the moving streak has been successfully integrated up out of the background by velocity filter processing. The measured output SNR was about 10 dB, which represents a 7-dB processing gain due to the 6-frame velocity filtering.

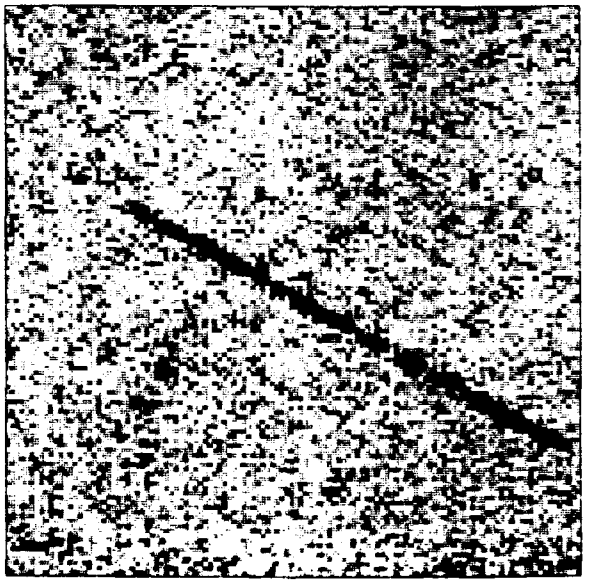
This example demonstrates the potential of the velocity filter to acquire very weak moving objects that could not be detected using ordinary single-frame thresholding or spatial filtering techniques.

4.5.2 Background Clutter Suppression. As explained in Section 4.1, the first step in optimum velocity filter processing is the suppression and normalization of non-homogeneous clutter. Clutter suppression experiments were conducted using the 11-frame sequence of MIT Quad-Camera 120 x 120 pixel images of the Great Nebula in Orion, which are plotted in Figure 24. The histogram of the 11th scene in the sequence is shown in Figure 25 to indicate the typical range of image pixel magnitudes prior to processing.

For optimum clutter suppression, we would ideally subtract out the nonstationary image mean from each pixel of every frame. Of course, in practice the true mean values are not known *a priori* and must be estimated from the available data. We tested two different methods of mean estimation; one



(a) Six Frames.



(b) Composit of Six Frames.

Figure 21. Telescope Images with Satellite Streak.

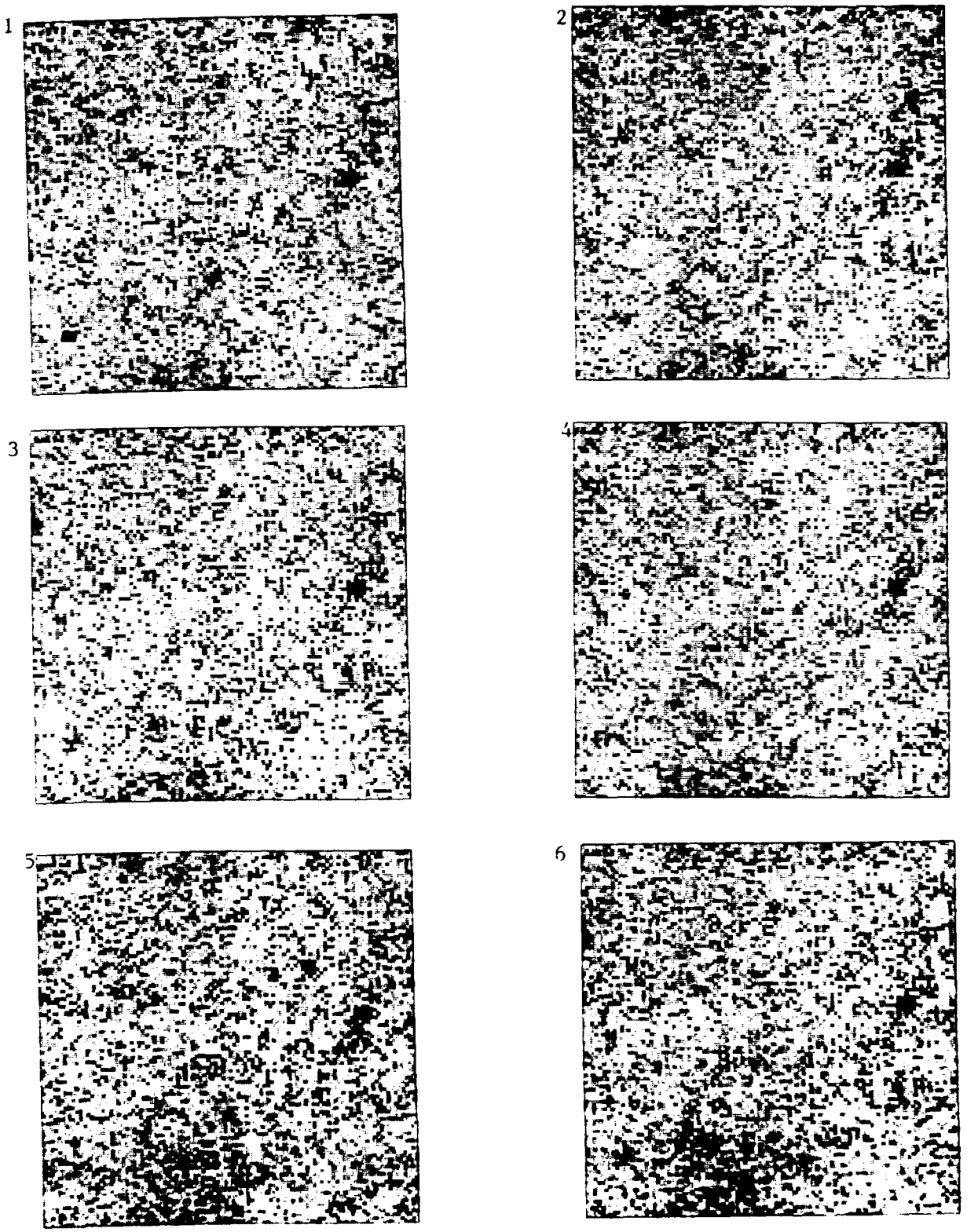


Figure 22. Images with Low Signal-to-Noise Ratio.

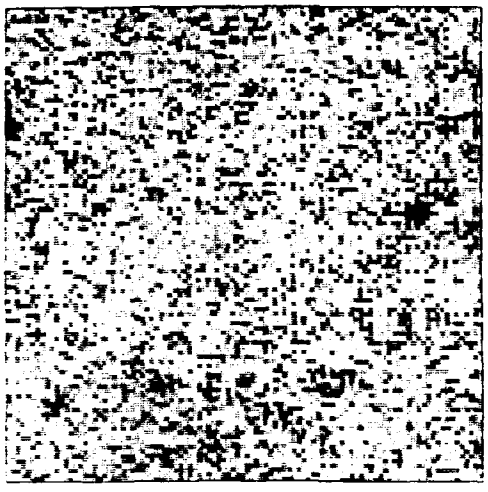


Figure 23a. Input Frame 6.

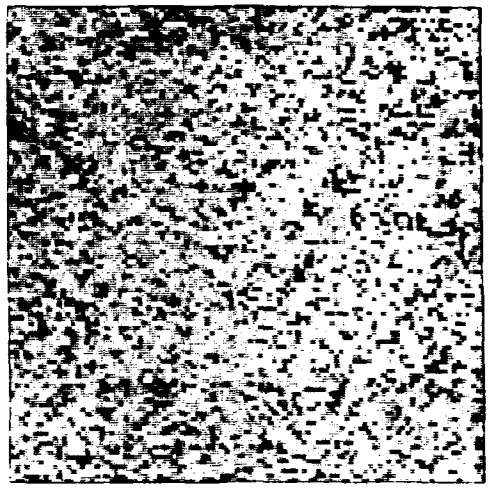


Figure 23b. Frame 6 After Median Removal.

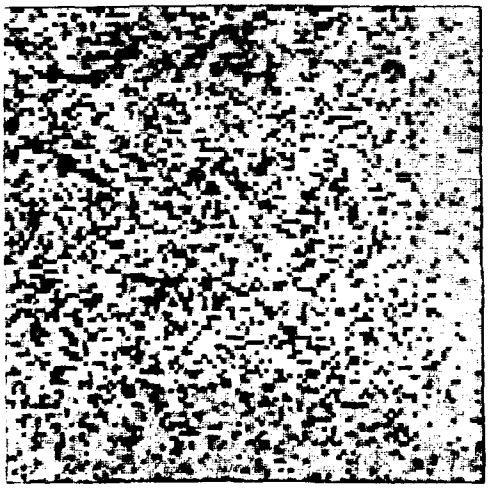


Figure 23c. Frame 6 After Shift-and-Add Velocity Filtering.

Figure 23d. Frame 6 After Correlation with Streak Function.

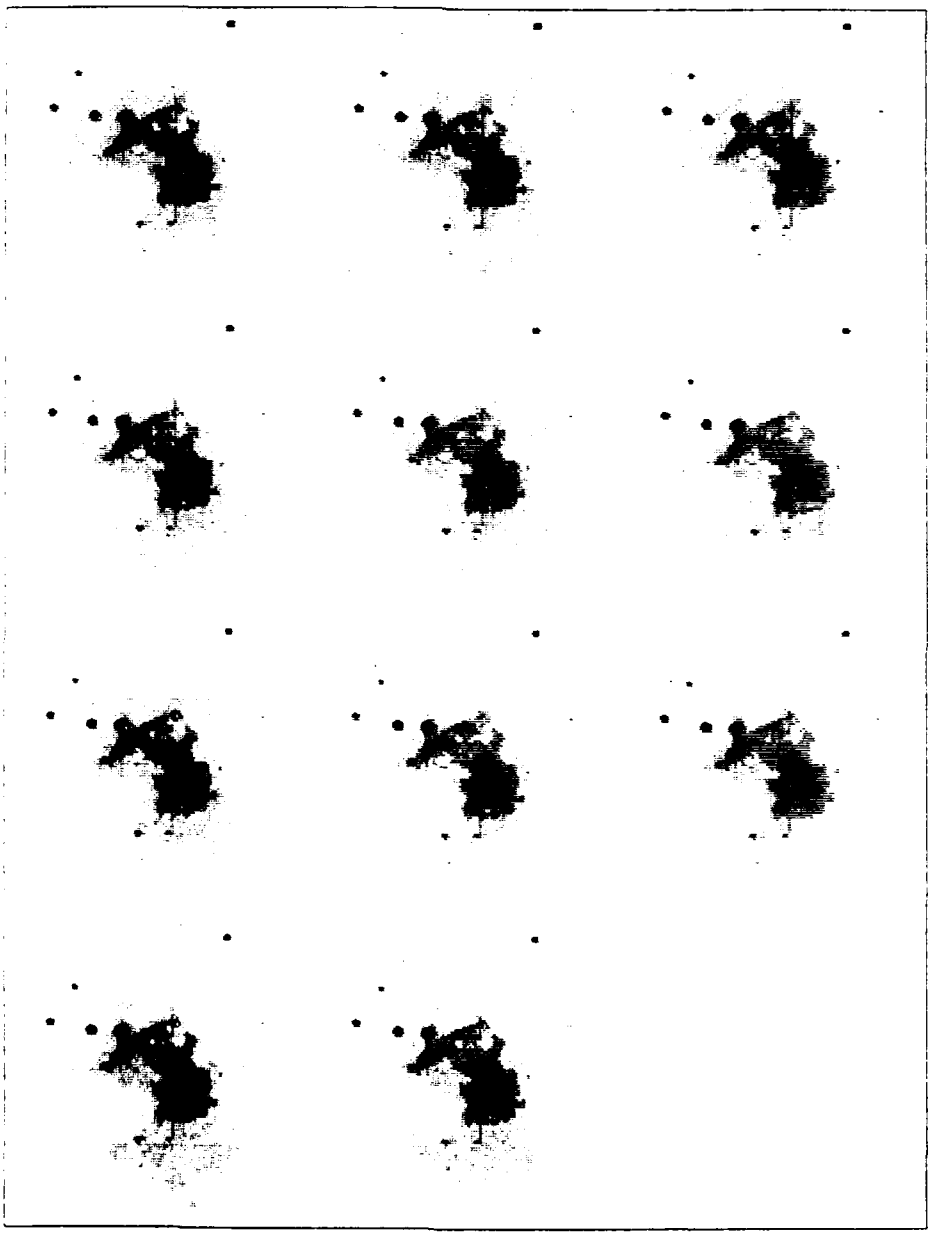


Figure 24. Telescope Images of the Great Nebula in Orion.

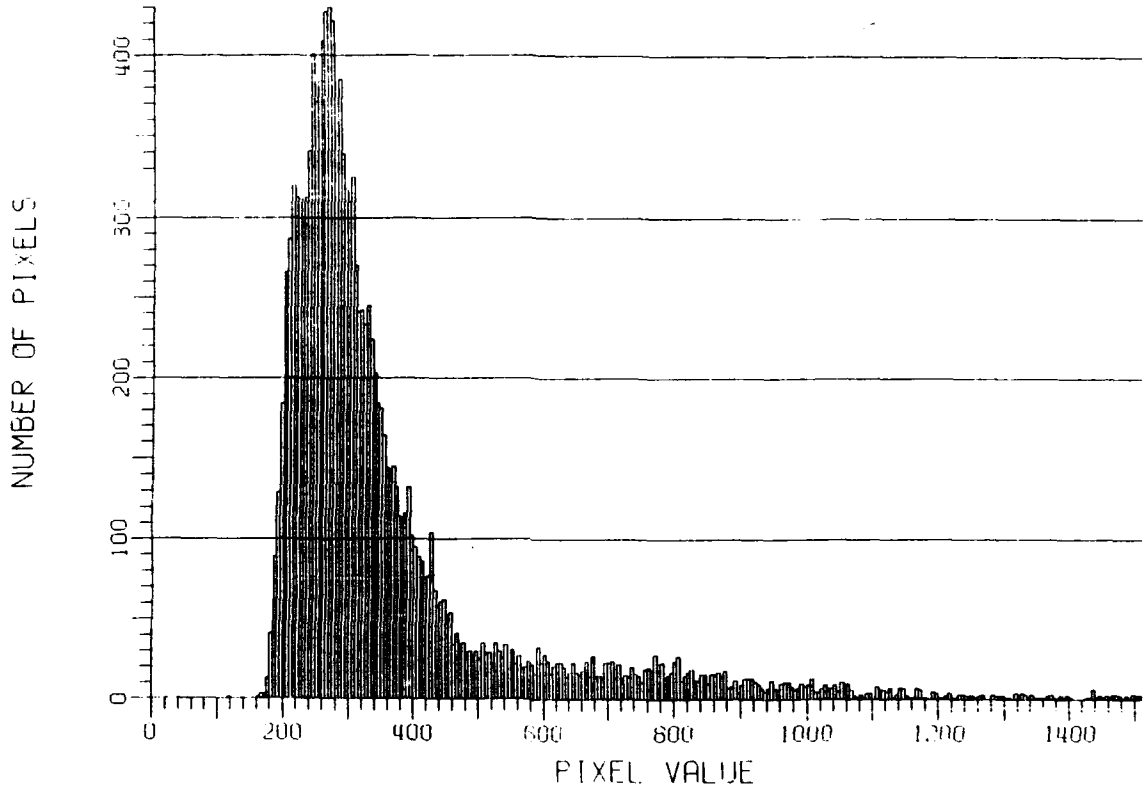


Figure 25. Histogram of Pixel Values in Frame 11 of Figure 24.

based on sliding-window spatial averages in each frame, and another based on time averaging across like-indexed cells of the different frames. The latter approach was found to be superior for the Orion dataset due to the extreme spatial variability of the background.

The results of mean removal applied to the raw data frames of Figure 24 are shown in Figure 26 (the dynamic range plotted here was increased by a factor of about 40 relative to the plots of Figure 24 in order to show sufficient image detail). The uneven results seen in the images of Figure 26 suggest that the original data frames are not correctly registered. We have empirically found that frame-to-frame jitter of even a small fraction of a pixel can greatly reduce the effectiveness of background suppression in structured clutter. Although in an operational system it might be possible to register successive scenes with measured data from an attitude sensor, in this case we had to self-register the scenes using cross-correlation techniques.

Figure 27 shows misregistration measurements that were obtained by cross-correlating the first frame with each of the others and precisely measuring the zero-offset of the correlation peak in both image dimensions. We then re-registered (or resampled) each original frame by passing it through a spatial interpolation filter having linear phase shifts proportional to the measured registration errors in each dimension. The results of mean removal on this new sequence of registered scenes are shown in Figure 28 (on the same gray scale used for Figure 26). The registration has eliminated most of the dead cells and readout artifacts (the vertical bands) seen in Figure 9, and the overall suppression is greatly improved. The small loss of image detail in Figure 28 was introduced by the low-pass characteristic of the spatial interpolation filter. Figure 29 shows the histogram of frame number 11 after registration and mean removal. The distribution of image magnitude is now centered on zero and the dynamic range of brightness has been considerably reduced from that shown in the original histogram of Figure 25.

The next stage of background suppression is a normalization by the estimated standard deviation σ on a pixel-by-pixel basis. The nonstationary σ was calculated as the square root of the unbiased sample variance in each pixel of the 11 frames. Figure 30 shows the 11 scenes after σ-normalization, and Figure 31 shows the modified histogram of the last frame. These figures show that the clutter suppression processing has transformed the highly structured background of the Orion scenes into an essentially homogeneous noise background with approximately Gaussian statistics. This operation is analogous to the "whitening prefilter" of linear matched filtering theory, whose role is to create a homogeneous Gaussian noise background against which the conventional matched filter detector is optimal.

4.5.3 Optimum and Suboptimum Velocity Filtering. The "optimum" velocity filter for an additive target moving over a spatially nonhomogeneous background is implemented via the following sequence of steps:

(a) *Background normalization:* Subtract the variable background mean from every pixel of each 2-D image frame, then normalize the result by dividing by the corresponding pixel variance;

(b) *Frame integration:* Shift the K background-normalized frames according to the target vector velocity (measured in pixels per frame) to create a set of frames in which the moving target's responses are superimposed, and add the corresponding pixels of these frames to create the velocity-filter output frame;

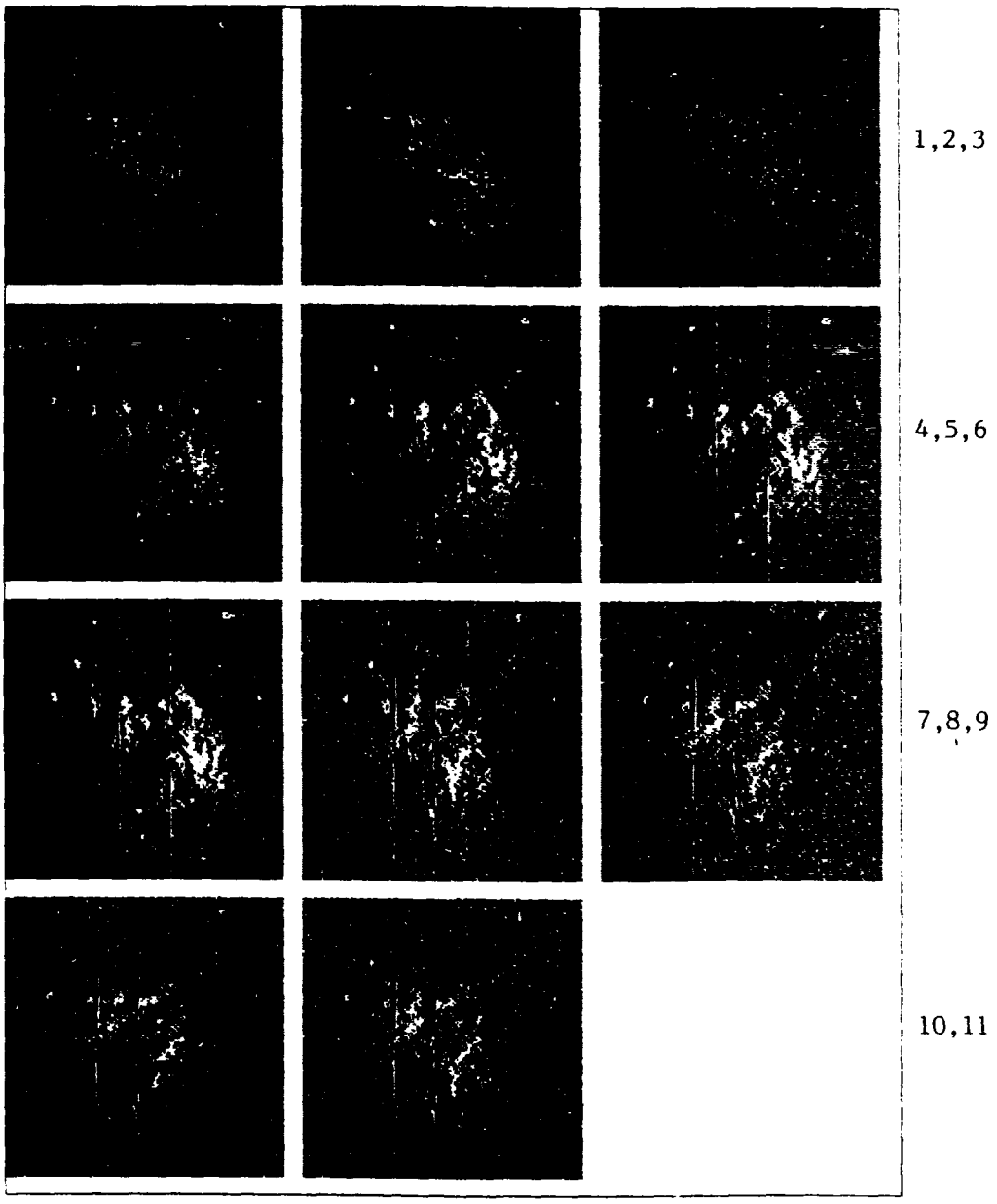


Figure 26. Orion Images with 11-Frame Mean Removed.

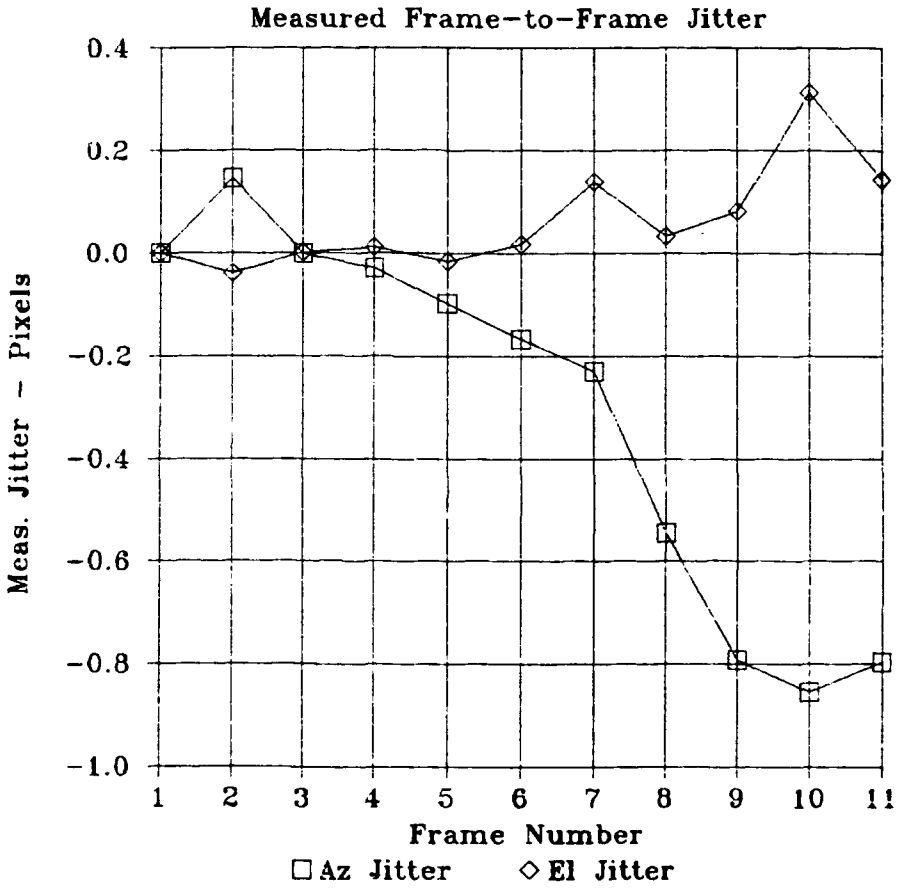


Figure 27. Misregistration Measurements Relative to Frame 1 for Orion Images.

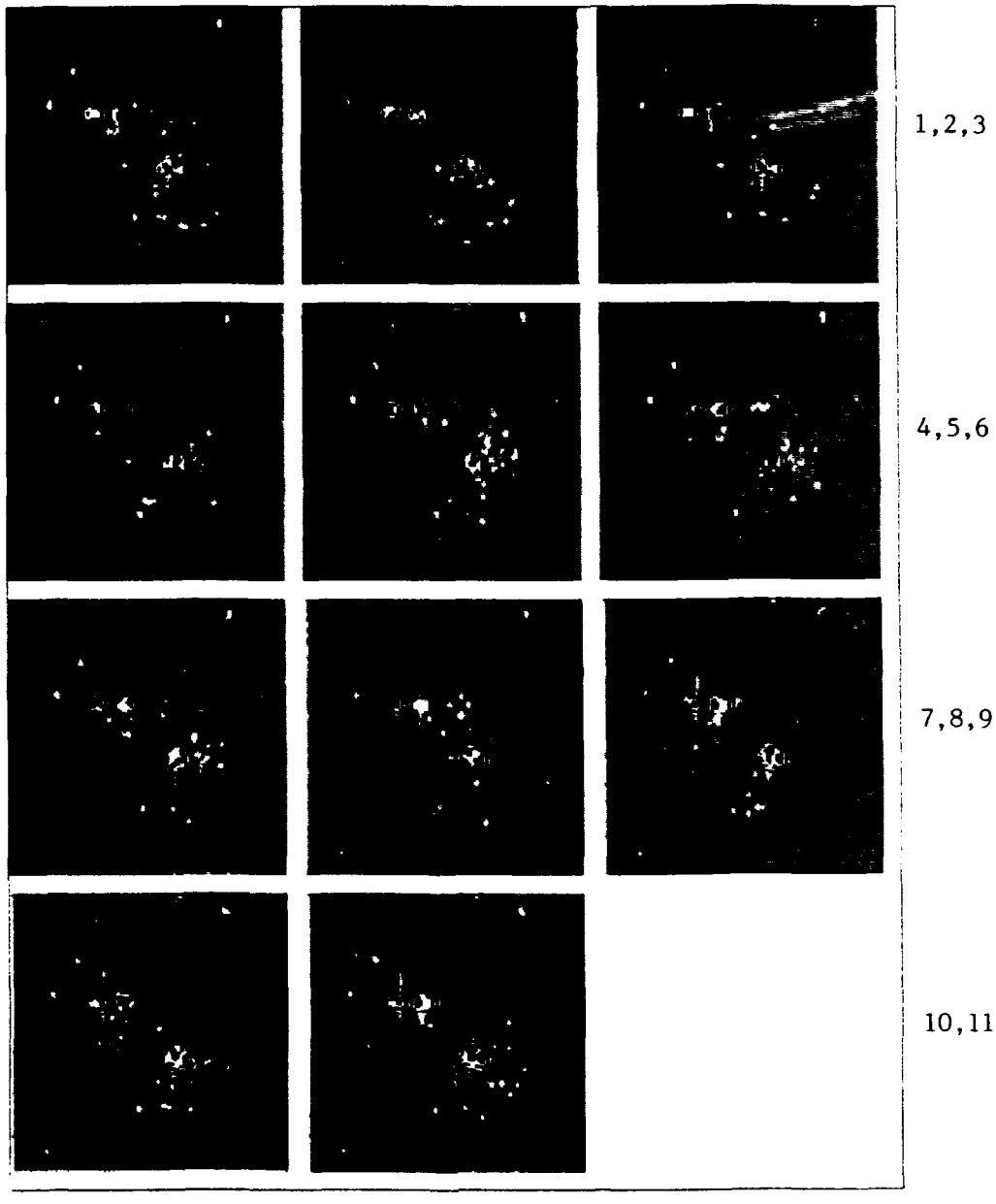


Figure 28. Registered Orion Images with 11-Frame Mean Removed.

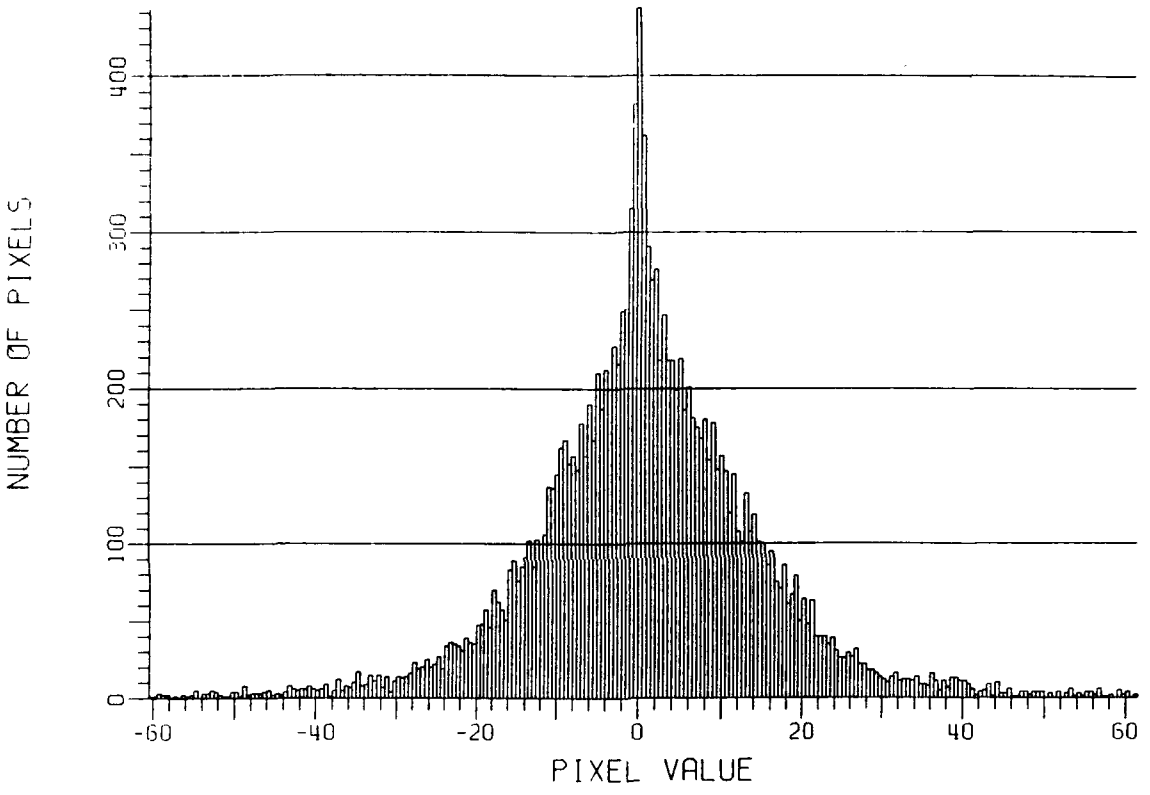


Figure 29. Histogram of Pixel Values in Frame 11 of Figure 28.

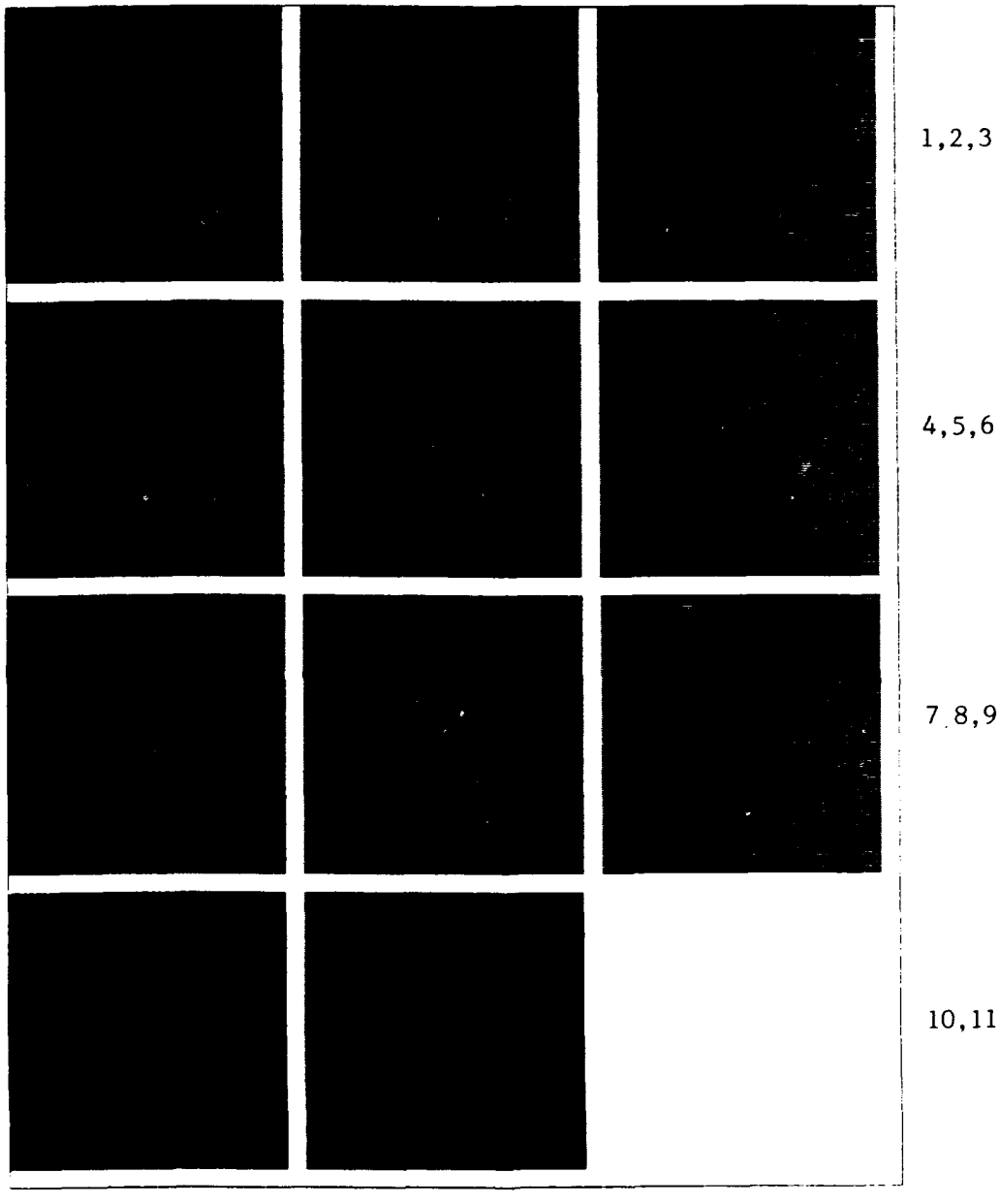


Figure 30. Registered Orion Images After Mean Removal and Normilazation by Standard Deviation.

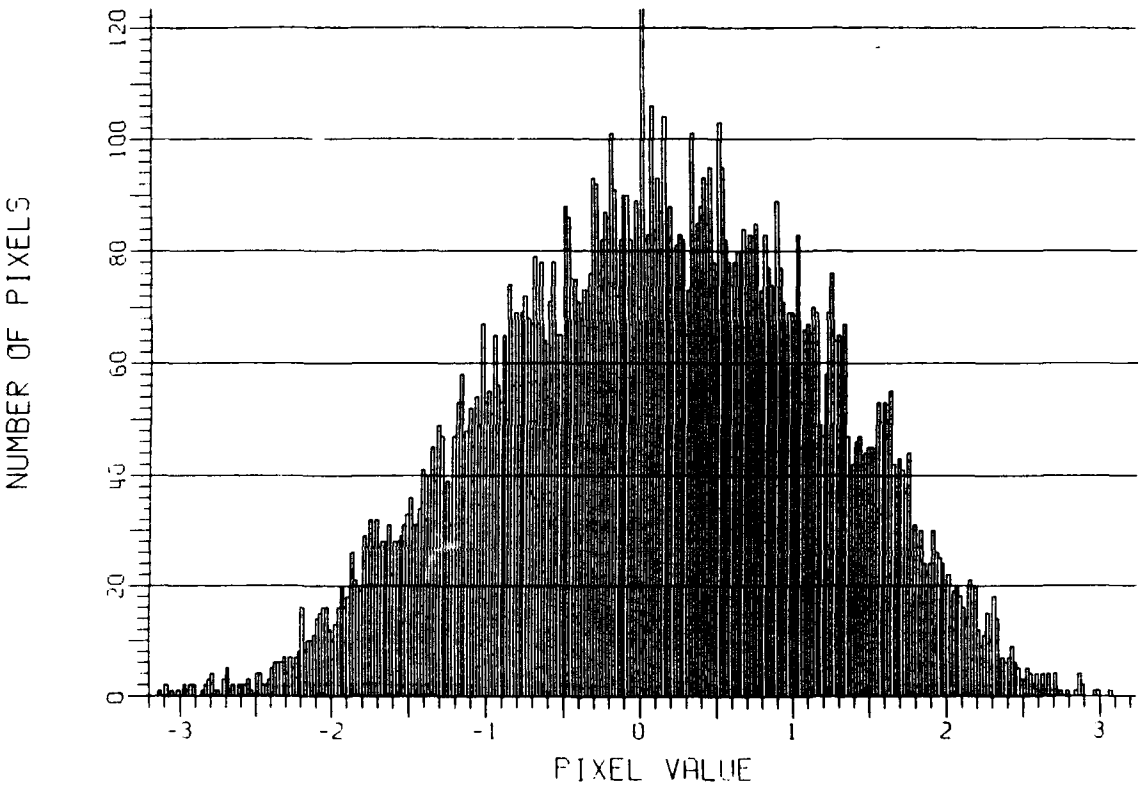


Figure 31. Histogram of Pixel Values in Frame 11 of Figure 30.

(c) *Spatial filtering* (for resolved targets which occupy multiple pixels): correlate the velocity-filter output frame with the target's spatial response;

(d) *Threshold* the result for detection and position measurement.

One practical disadvantage of this optimum procedure stems from the fact that the variance of the pixel fluctuations in not the same in each pixel. (This is because in the background normalization step we divide by the variance rather than by the standard deviation.) The result is that the threshold required to maintain a given false alarm rate P_{FA} (for non-homogeneous clutter) varies with the signal shape and with the target's path through the K frames. The need to compute a different threshold for each pixel of the velocity-filter output frame for every velocity filter doubles the processing required. This has led us to consider a suboptimum velocity filter in which the normalization by division by the pixel variance is replaced by division by the pixel standard deviation. In that case the velocity-filter output variance is the same in each pixel, even for a non-homogeneous background, and a single threshold can be used for every pixel of every velocity-filter output frame. However, this simplification is obtained at the expense of a nominal loss in detection performance. The amount of this loss depends on the degree on non-homogeneity in the background pixels over which the target moves.

The average input amplitude signal-to-clutter ratio (SCR) for a non-constant background is given by

$$SCR_{in} = \sqrt{\frac{1}{K}\sum_{k=1}^{K}\frac{A^2}{\sigma_k^2}} \tag{14}$$

where K is the number of frames to be velocity filtered, A is the amplitude of the (assumed additive) target, and σ_k^2 is the background variance in the pixel occupied by the target in frame k. The SCR at the output of the optimum velocity filter (division by the variance in each pixel) is

$$SCR_{out} = \sqrt{K}\cdot SCR_{in} = \sqrt{\sum_{k=1}^{K}\frac{A^2}{\sigma_k^2}}, \tag{15}$$

while the SCR at the output of the sub-optimum velocity filter (division by the standard deviation in each pixel) is

$$SCR_{out} = \frac{\sum 1/\sigma_k^2}{\sqrt{\sum 1/\sigma_k^2}}\cdot SCR_{in} = \frac{1}{\sqrt{K}}\sum_{k=1}^{K}\frac{A}{\sigma_k}. \tag{16}$$

Several experiments were performed on the Orion dataset to evaluate the effectiveness of these two forms of background suppression and velocity filtering. In one case we injected a set of 10 relatively weak constant-amplitude, blurred point targets in two distinct velocity clusters into the 11 image frames; their

initial positions and trajectories are shown superimposed on the background in Figure 32. Figure 33 shows the positions of the target responses in each of the 11 frames. Since each target traversed a different portion of the Great Nebula scene, the average input signal-to-clutter ratio (SCR) was different for each target, as indicated in Table I.

Table I. Measured Performance of 11-Frame Velocity Filters

Velocity Cluster	Target Number	Average Input SCR	Avg. Output SCR (Optimum Filter)	Avg. Output SCR (Suboptimum Filter)	SCR Loss
1	1	6.04 dB	16.46 dB	15.68 dB	0.78 dB
1	2	6.60	17.01	16.63	0.39
1	3	-0.42	10.00	9.48	0.52
1	4	2.30	12.71	11.53	1.18
2	1	7.70	18.12	17.58	0.54
2	2	5.42	15.83	14.94	0.89
2	3	7.20	17.62	17.30	0.32
2	4	6.55	16.96	16.67	0.29
2	5	4.66	15.08	14.79	0.29
2	6	7.29	17.70	17.35	0.35

Figure 34 plots the 11 input frames (after image registration) with the synthetic targets included. The optimum background suppression (division by the pixel variance) was then carried out on each frame, and the resulting average output SCR for each target was as shown in the fourth column of Table I. The optimum filters produced an average SCR gain of $\sqrt{11} = 10.41$ dB for each target. We also tested the sub-optimum (division by the pixel standard deviation) background suppression. The last column of Table I shows that in this case the output SNR was 0.3 to 1.2 dB lower than for the optimum, depending on the particular target.

The output frame for each of the two suboptimum velocity filters is shown in Figures 35a and 35c. Application of a fixed threshold to these frames (at a false alarm probability of around 10^{-6}) produced five detections as shown in Figures 35b and 35d. The five detected targets had measured output SNR levels in the vicinity of 16 to 17 dB, while the undetected targets (with the unlucky exception of target 1 in cluster 2) had marginal SNRs from 1 to 6 dB lower. To reliably detect the remaining targets, we could integrate more frames in the velocity filters, or require a higher input SNR for those targets. Figure 36 shows the results of suboptimum velocity filtering and thresholding for a case where the power of each of the injected objects was 10 dB higher on each frame. In this case, all 10 point targets were easily detected against the suppressed clutter background.

4.5.4 Multiple-Target Track Initiation. So far, we have emphasized the problem of detecting weak objects in a severe clutter and noise background. This is the original application for which the velocity filter was designed. However, we have also found that the velocity filter concept can be applied to the problem of track initiation from a set of target detections obtained from multiple sensor scans. The velocity filter appears to be especially well-suited to this function in typical midcourse strategic defense

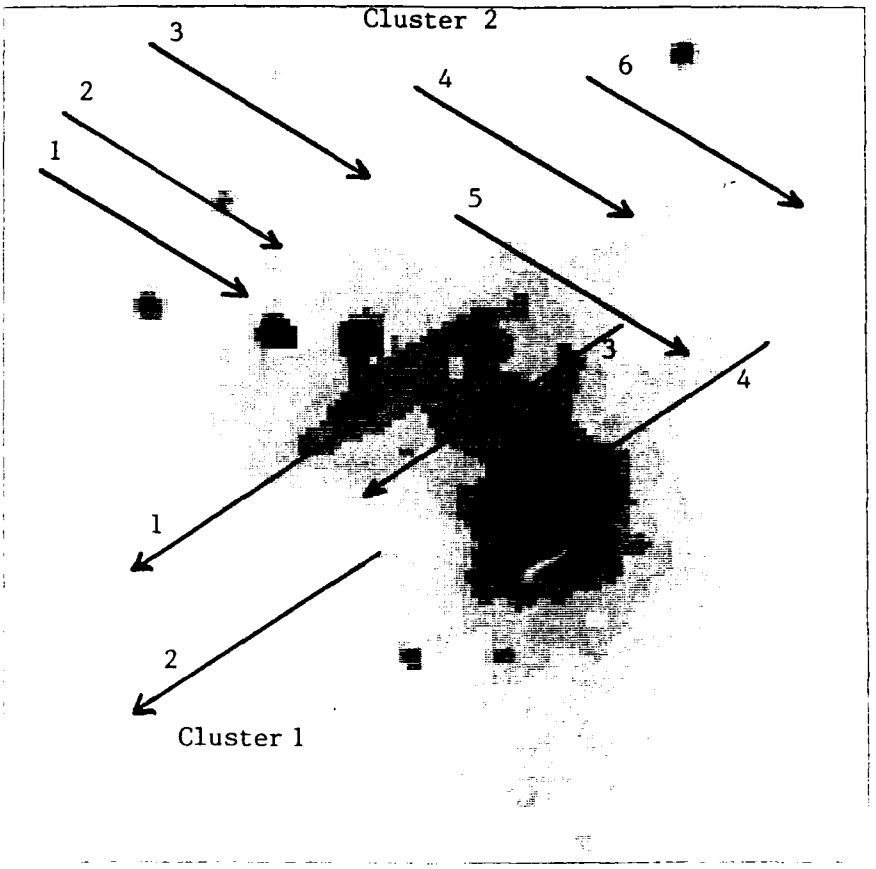


Figure 32. Initial Positions and Trajectories of 10 Targets in Two Velocity Clusters.

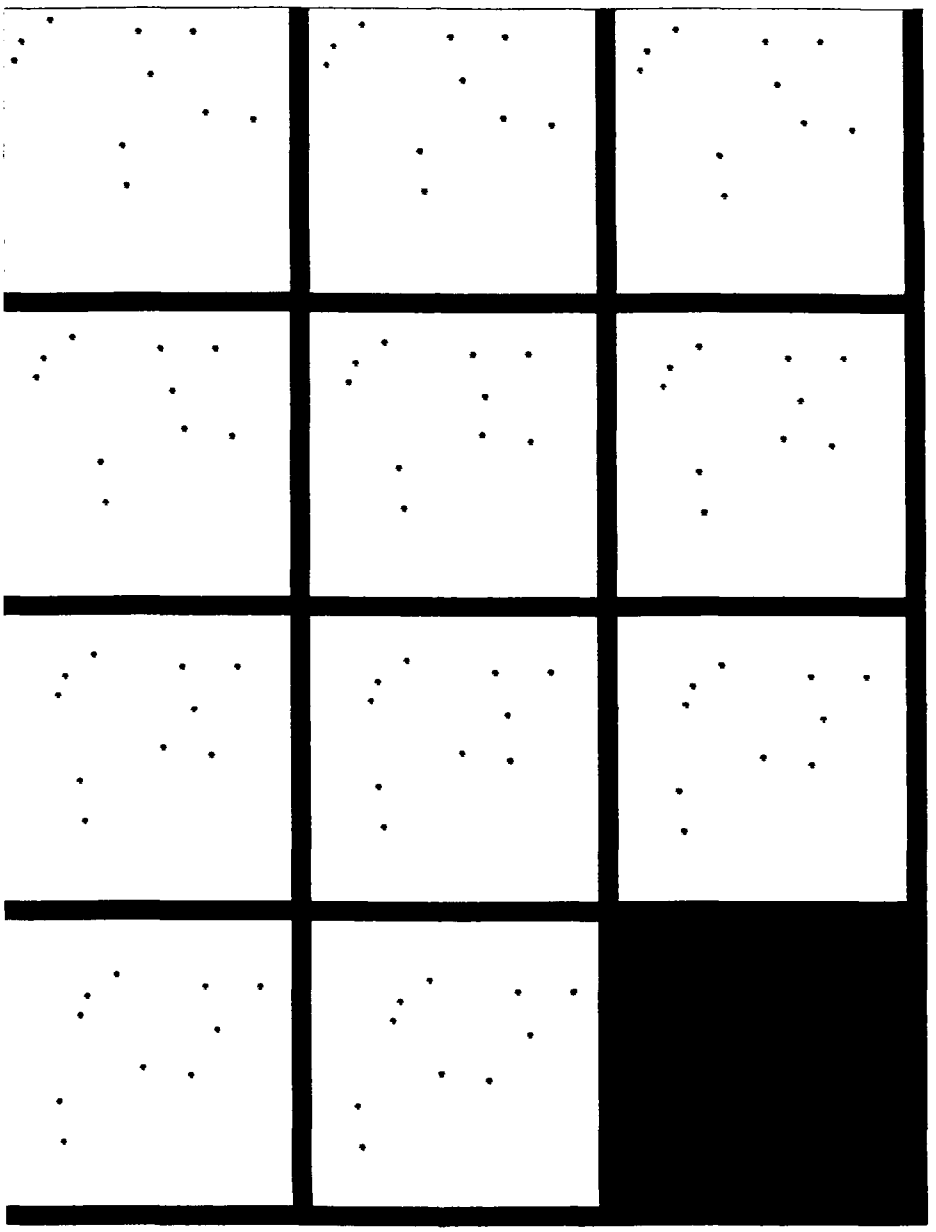


Figure 33. Positions of the Point Target Responses in 11 Frames.

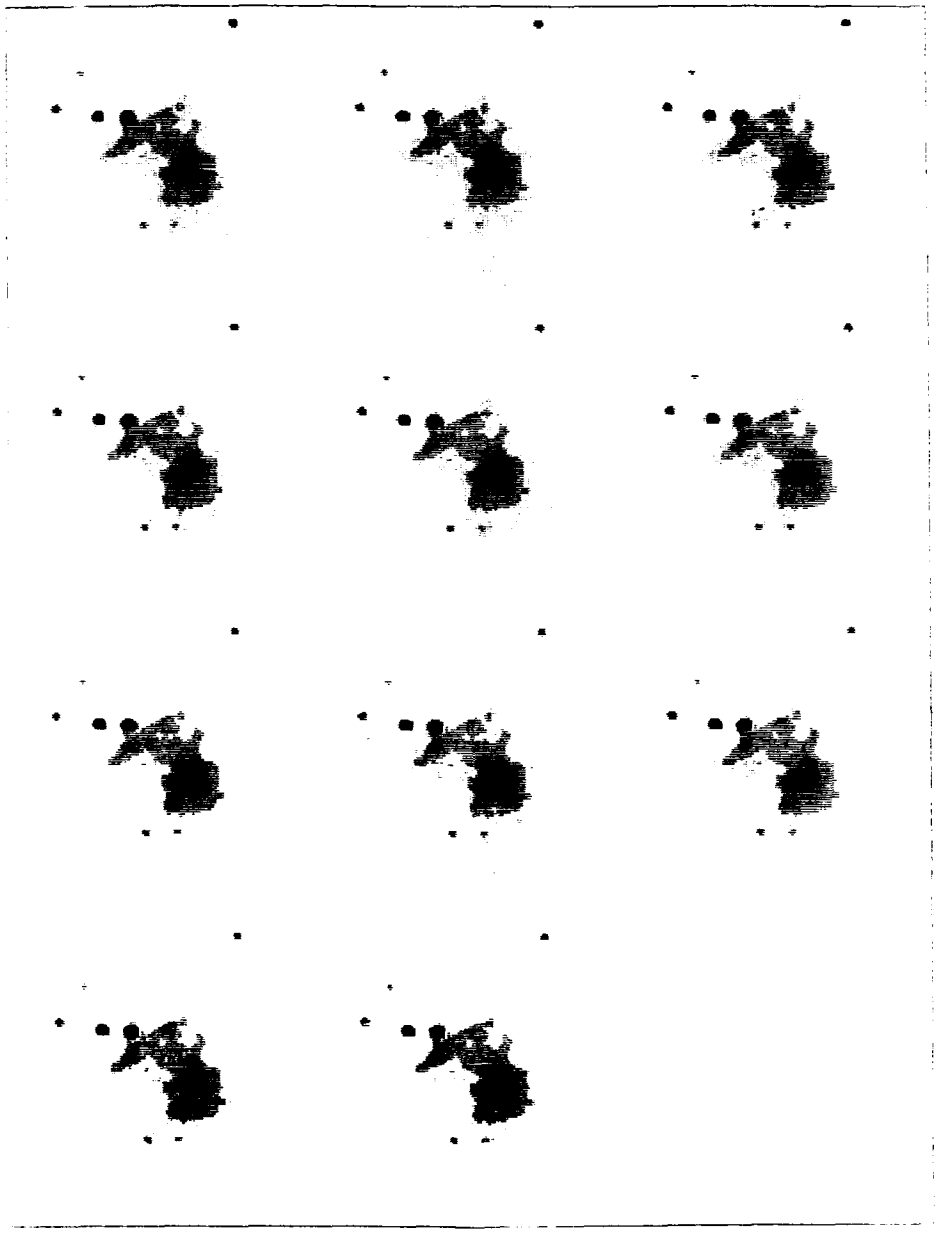


Figure 34. Registered Orion Image Frames with Synthetic Targets Injected.

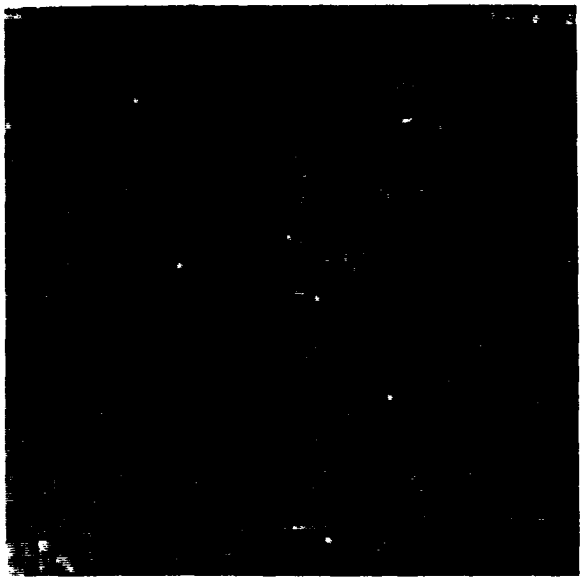


Figure 35a. Velocity Filter Output for Target Cluster 1.

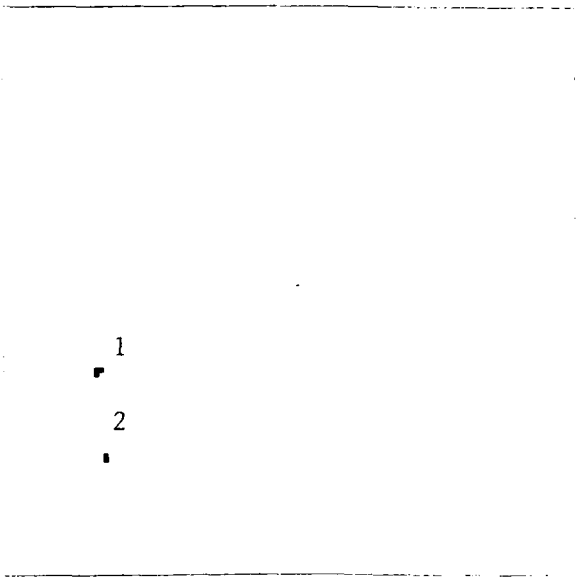


Figure 35b. Detections Obtained from the Output of Figure 35a.

Figure 35c. Velocity Filter Output for Target Cluster 2.

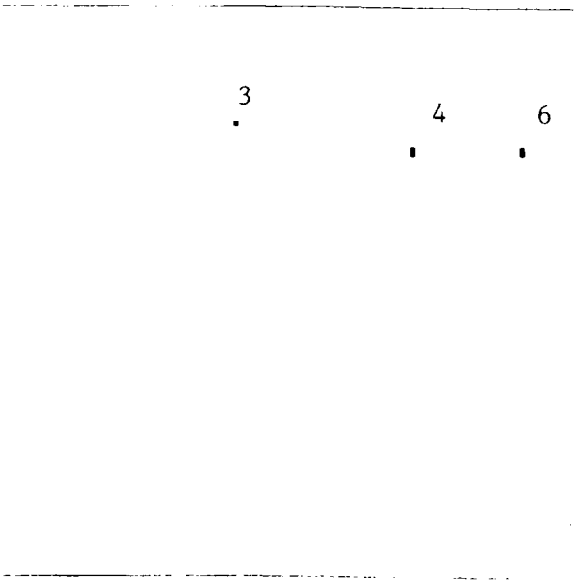


Figure 35d. Detections Obtained from the Output of Figure 35c.

Figure 36a. Velocity Filter Output for Target Cluster 1 (10 dB Higher SNR).

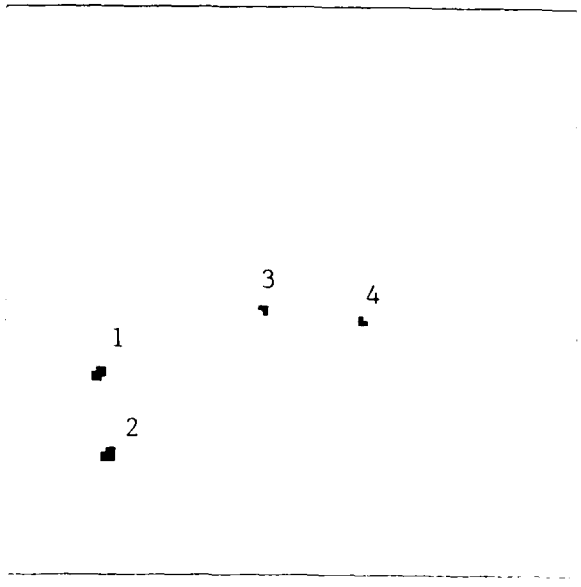


Figure 36b. Detections Obtained from the Output of Figure 36a.

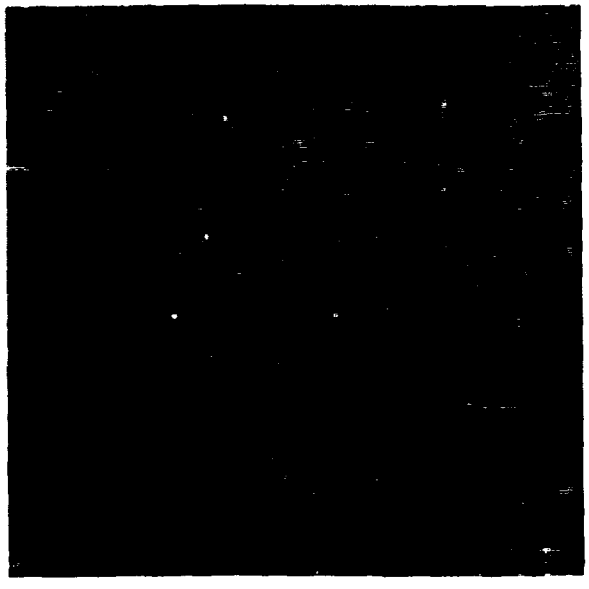


Figure 36c. Velocity Filter Output for Target Cluster 2 (10 dB Higher SNR).

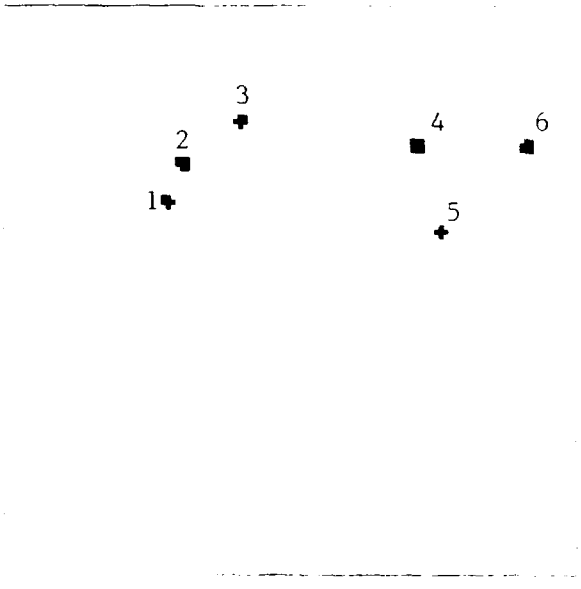


Figure 35d. Detection Obtained from the Output fo Figure 36c.

encounters. Moreover, it leads to a novel "signal processing" interpretation of the group track initiation problem.

Figure 37 illustrates the physical basis for the velocity filter approach as applied to midcourse target acquisition. These plots show apparent angular position and velocity "truth" for several clusters of objects viewed by a probe sensor at two different times in a simulated midcourse encounter generated by the U.S. Army Strategic Defense Command. Each cluster arises from the deployment of multiple objects (reentry vehicles and decoys) with relatively small differential velocities from a single post-boost vehicle. The clusters tend to expand slowly and may merge and cross as time progresses due to the cumulative effects of these initial velocity differences and the non-uniformity of the earth's gravitational field.

The presence of (possibly unresolved) target clusters in *position space* has been widely recognized as an important constraint of the midcourse tracking problem; group tracking approaches that exploit this clustering have been developed [17,18]. The velocity filter suggests an alternative group tracking concept which exploits the natural clustering of objects in apparent *velocity space*. When viewed in velocity space, midcourse objects tend to remain tightly clustered in stable patterns for much longer time periods than the corresponding position clusters (see Figure 37). This can greatly simplify the central problem of track initiation: the establishment of a frame-to-frame correspondence between object clusters. If this correspondence problem can be easily solved in velocity space, then the velocity histories of individual clusters in the field-of-view are immediately available. A velocity history makes it possible to construct a filter matched to the motion of a given cluster. The thresholded output of the filter reveals the angular positions within the frame of resolved objects in the cluster, thereby achieving the equivalent of group track initiation. Since both the velocity measurement and filtering operations can be performed by bulk signal processing, and since there are typically many objects per cluster, this procedure can be highly efficient from a computational point of view.

Velocity Filter Track Initiation Concept. Figure 38 shows a basic block diagram of the velocity filter approach to cluster track initiation. The input is a sequence of image frames, each of which consists of threshold exceedances obtained as a function of apparent azimuth and elevation during a scan period. These frame are first cross-correlated with one another to obtain measurements of the apparent cluster velocities at various times. If necessary, the peaks in the cross-correlation functions are used to track the cluster velocities and thereby obtain initial estimates of higher-order derivatives such as acceleration. Next, the measured motion characteristics of the clusters in the field-of-view are used to build a bank of filters matched to the objects in various clusters. Each filter processes the input frame sequence to produce a single output frame in which the intensities of objects in a given velocity cluster are enhanced relative to those of all other objects. This output frame is then thresholded to yield position estimates for all target responses within its passband.

Frame Generation for a Simulated Threat. The specific steps involved in this approach have been demonstrated by processing a 112-object midcourse data set generated by the M.I.T. Lincoln Laboratory. This threat actually represents a somewhat stressing case since the object clusters are not particularly well resolved and have significant higher-order apparent motion.

Figure 39 shows four frames generated from this simulated trajectory data for a scanning sensor with a 10 second scan period. The frames were created by placing a 2-D Gaussian response (with unit amplitude and a 1σ blur circle radius of 50 μrad) at the apparent angular location of each detected threat

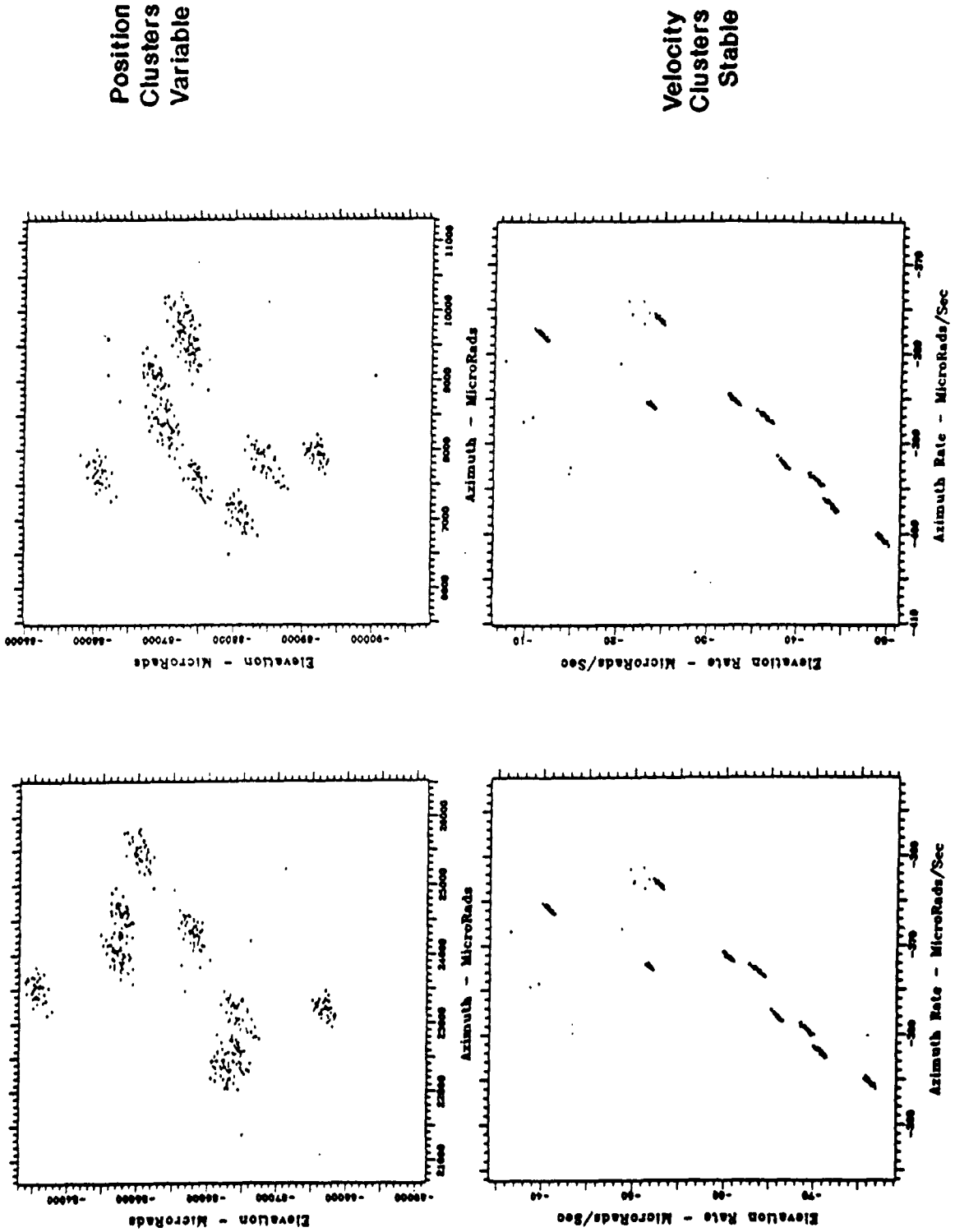


Figure 37. Physical Basis for Velocity-Filter Approach.

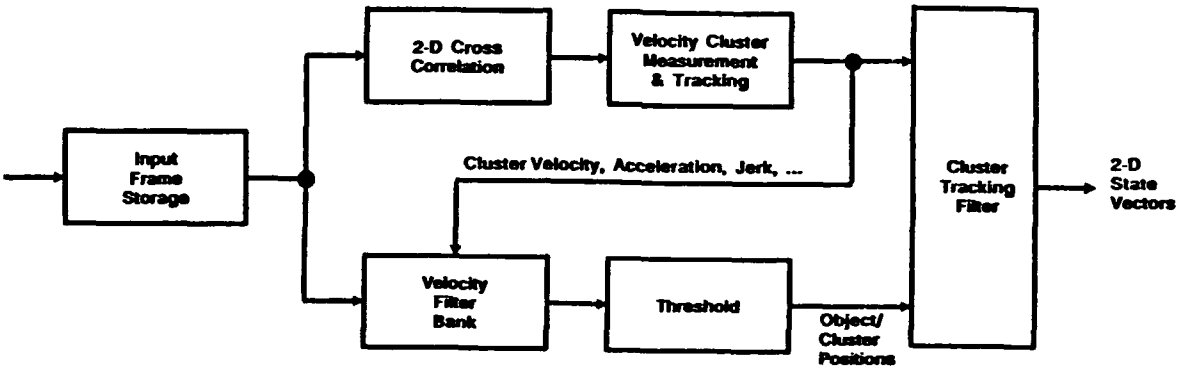


Figure 38. Velocity-Filter Track Initiation Concept.

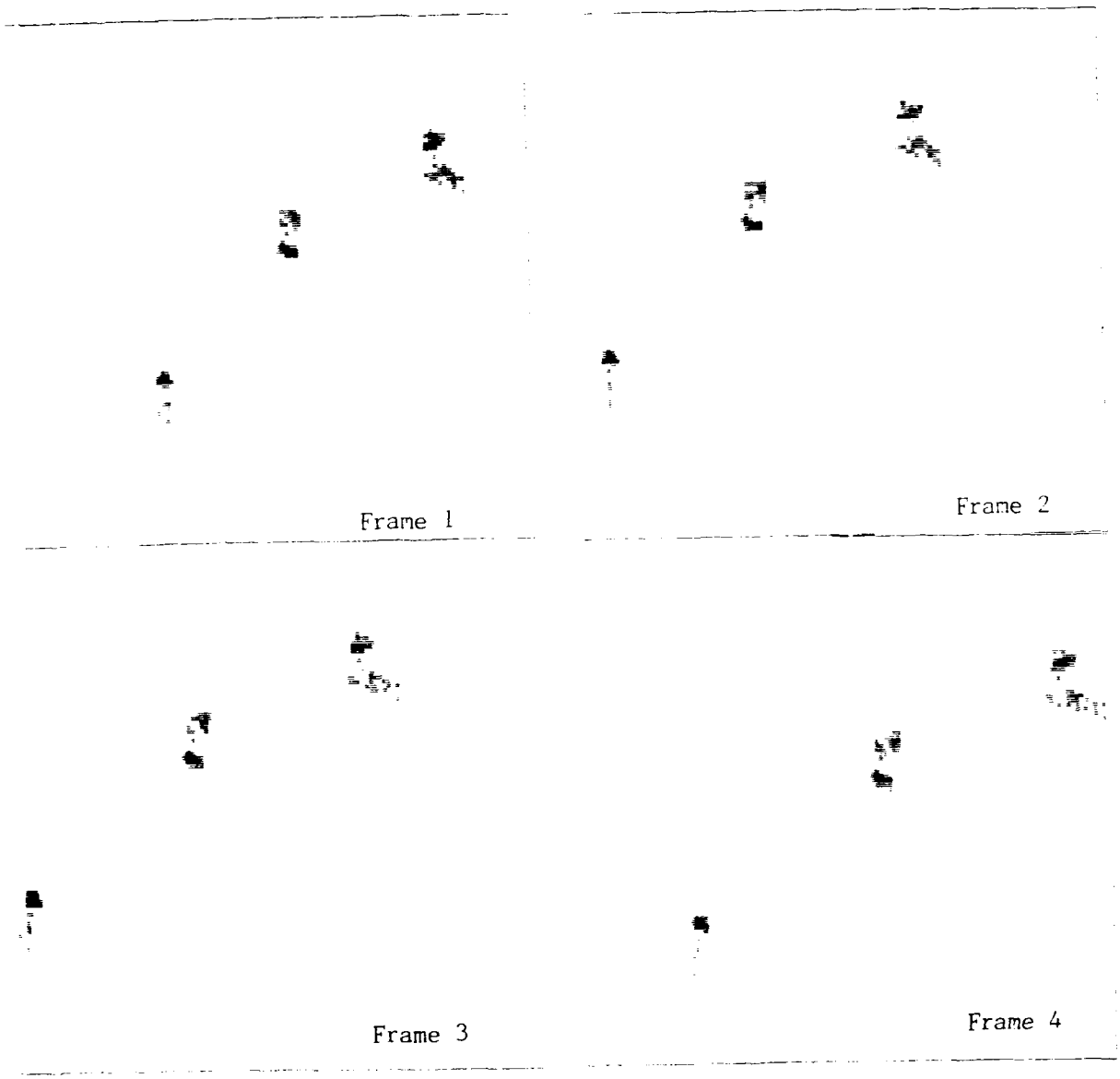


Figure 39. Four Input Scenes.

object. Each frame thus represents a set of threshold crossings that might be obtained by a passive sensor viewing these objects during a single scan period. Note that some of the target responses in each cluster appear blurred together in Figure 39 because they are too closely-spaced to be resolved by the hypothetical passive sensor.

Frame Cross-Correlation. Since apparent velocity cannot be directly observed by a conventional passive sensor, the frame data must be pre-processed to generate such measurements. This is accomplished by computing the cross-correlation functions for all possible pairs of frames, as shown in the six contour plots of Figure 40. The functions are plotted on a common normalized scale of pixels/frame in the azimuth and elevation dimensions. The highest peak found in each function is denoted by an arrow. The location of this peak gives an estimate of the group velocity averaged over the time separating the two frames. Note that the location of the highest peak changes from one plot to another, indicating that the moving objects in these frames have significant apparent acceleration.

Velocity and Acceleration Measurement. Figure 41 shows the measurements of normalized azimuth and elevation rate obtained from the locations of the peaks indicated in the plots of Figure 40. The time associated with each group velocity measurement is taken as the midpoint of the two times associated with the frame pair used to form the corresponding cross-correlation function. Estimates of the group acceleration in each dimension are then obtained by fitting straight lines through the respective sets of measurements shown in Figure 41, and measuring their slopes. In this particular example, all of the velocity measurements are found to lie very close to the least-squares lines, indicating that higher-order motion components beyond the acceleration term are insignificant.

To proceed with velocity measurement, the 2-D group acceleration estimate is used to shift the correlation functions of Figure 40 to compensate for the change in group velocity from frame-to-frame. Contour plots for these shifted correlation functions are shown in Figure 42 on a velocity scale referenced to the time of the first frame. The final step in velocity estimation is to combine the six cross-correlations of Figure 42 in what basically amounts to a coincidence detection operation. Specifically, the functions are combined by limiting all values below a selected threshold to zero and computing the geometric mean; that is, multiplying them together and taking the sixth-root. This operation effectively eliminates the "ghost" peaks which result when one cluster partially correlates with a different one in the correlation function of highest resolution (the one for frames 1 and 4).

The central portion of this combined correlation function is plotted in contour format in Figure 43. The peak to the left of the plot is well-defined and represents the velocity of one of the object clusters at the first frame. The larger response to the right has two subsidiary peaks which are fairly close together. These peaks indicate the velocities of two other clusters, which are marginally resolved in velocity space.

Velocity Filtering. Since three clusters can be detected and measured in velocity space, three different velocity filters are implemented on the input frames shown in Figure 39. For a constant-velocity target whose frame response can be approximated by a 2-D Gaussian function of radius σ_b, it was shown in Section 4.3.2 that the half-power velocity passband of a shift-and-add velocity filter is

$$\Delta v_{3dB} = 6\sigma_b/KT \qquad (11)$$

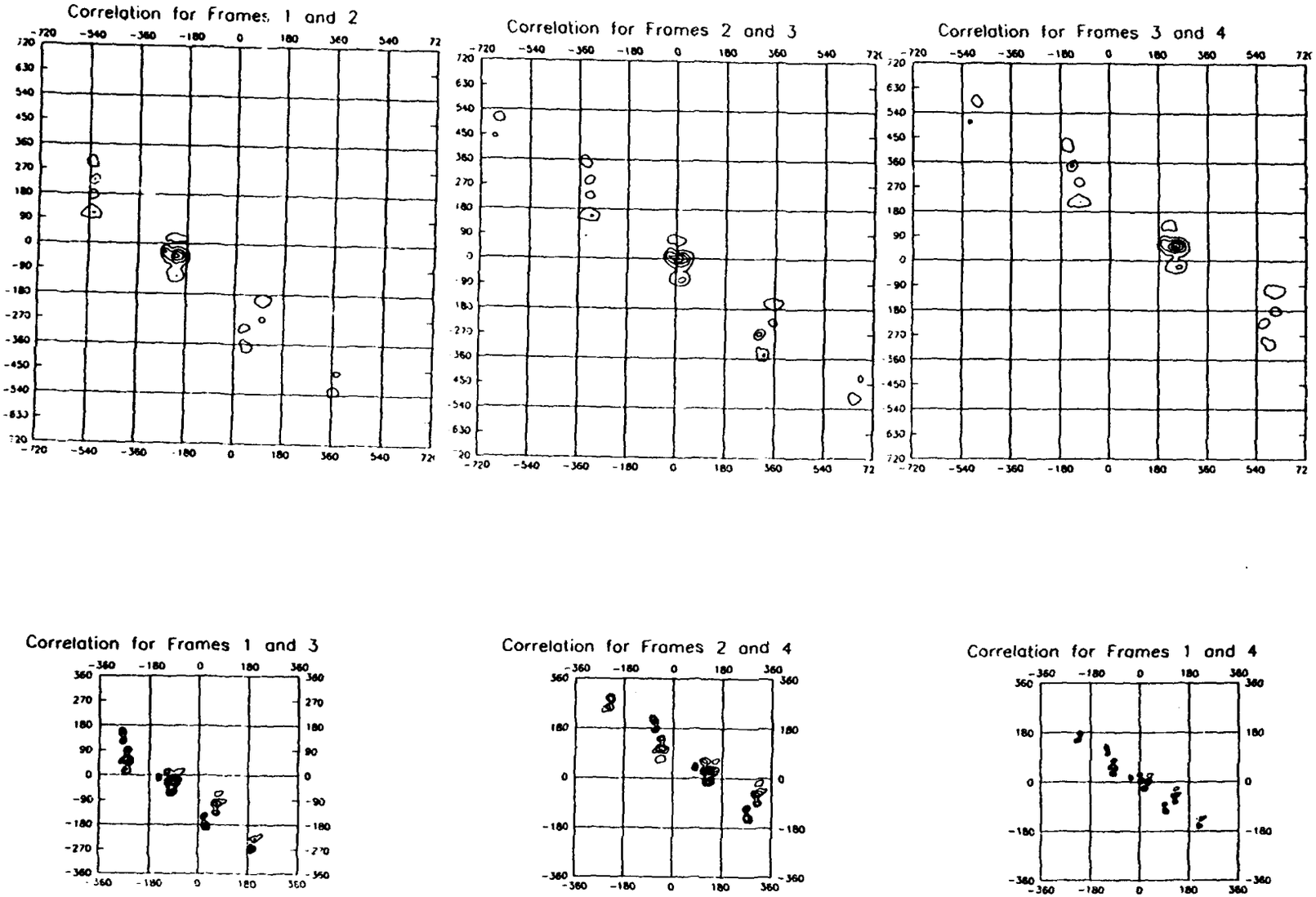


Figure 40. Cross Correlations Scaled as Apparent Velocities.

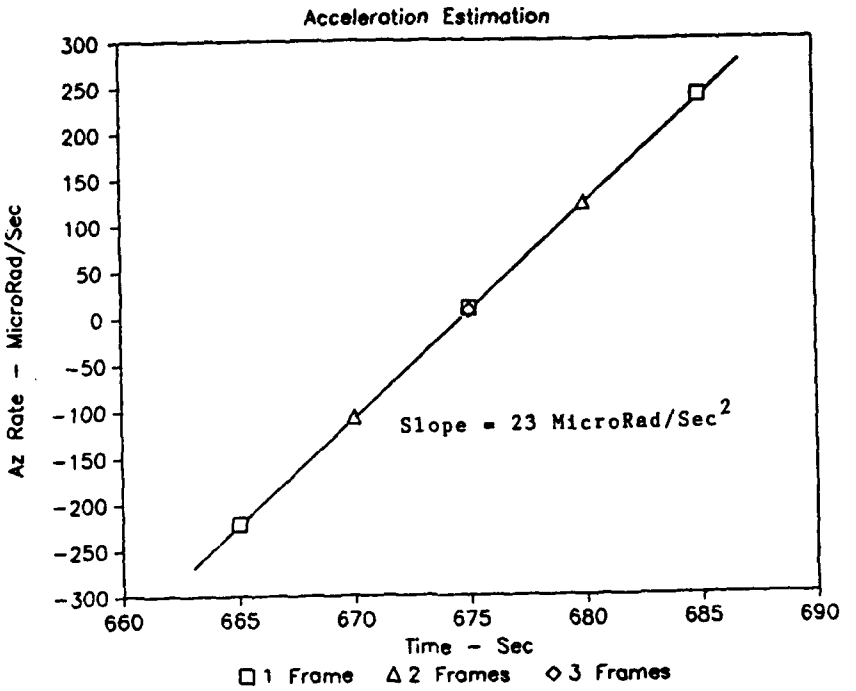


Figure 41. Azimuth Acceleration Estimate from Correlation-Peak Locations.

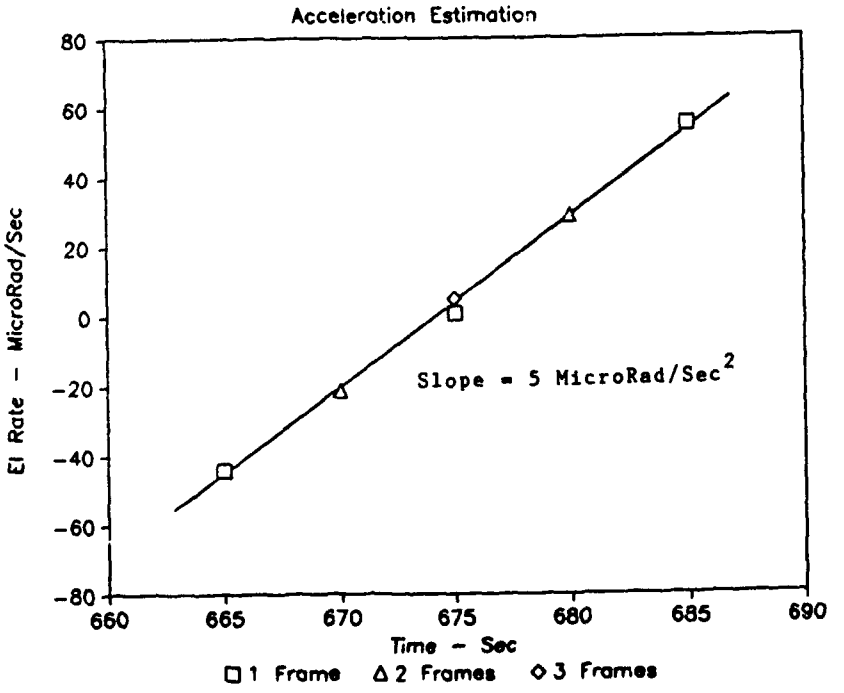


Figure 41b. Elevation Acceleration Estimate from Correlation-Peak Locations.

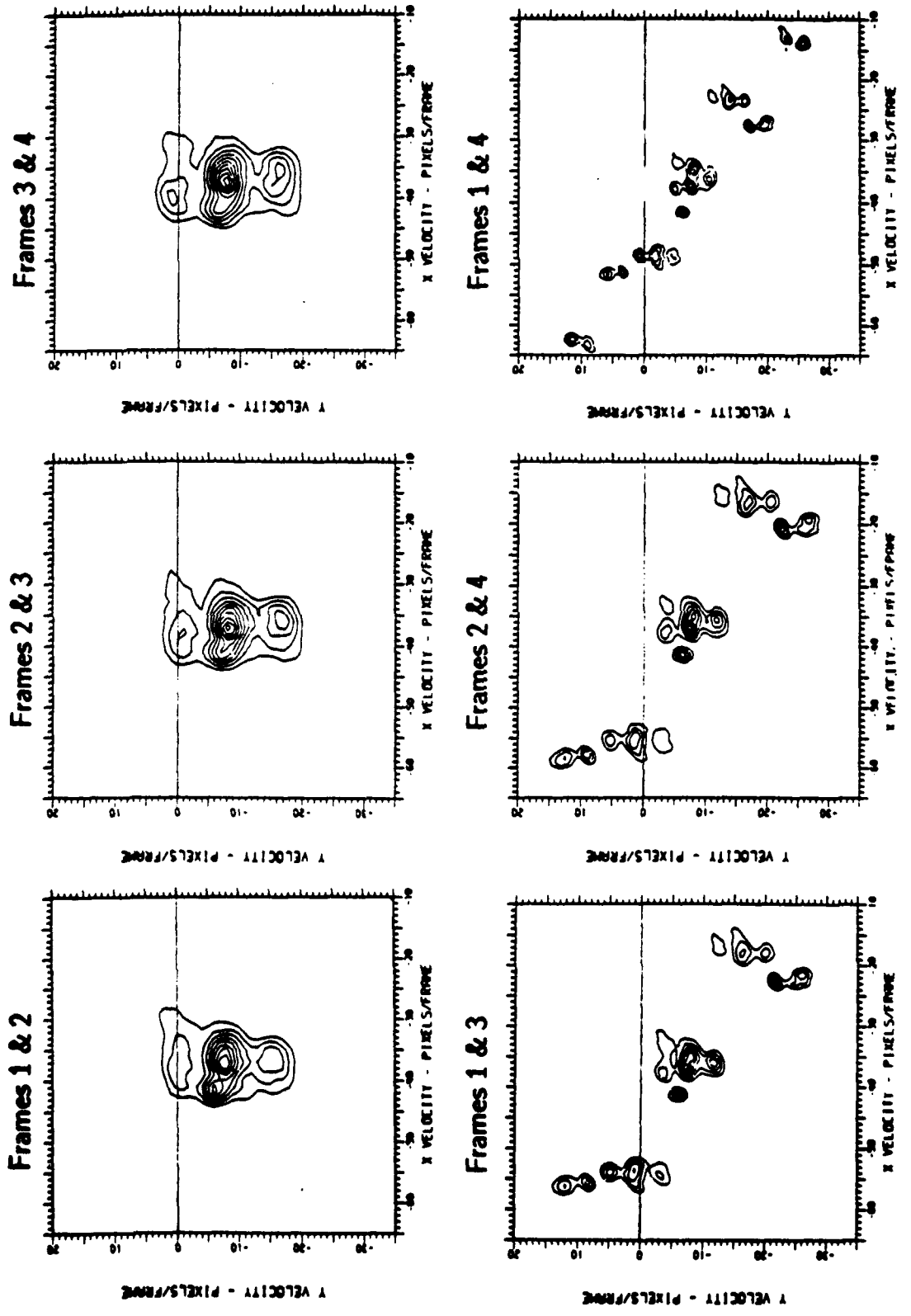


Figure 42. Correlation Functions After Shifting for Acceleration.

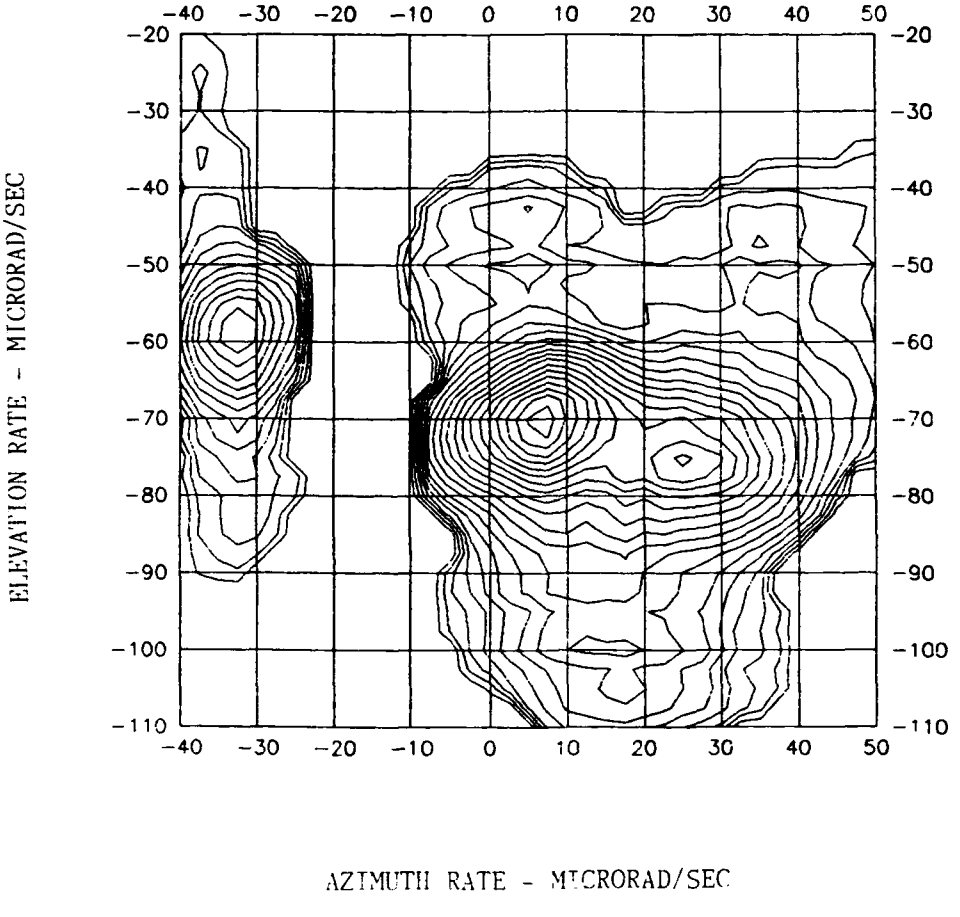


Figure 43. Geometric Mean of Shifted Correlation Functions.

where K is the number of frames filtered and T is the frame interval. In this example we have a blur radius of σ_b=50 μrad, N=4 and T=10 sec, so the highest expected velocity resolution for an isolated object is Δv=7.5 μrad/sec in either dimension. (This of course assumes that any measured acceleration components are properly accounted for in the shift-and-add process.) However, the actual resolution against clustered objects may be somewhat less than this in the present case, since the effective angular extent of some of the CSO responses exceeds that of the blur circle.

Figure 44 shows the final output of the three velocity filters after thresholding. These filter outputs are referenced to the time of the last of the four input frames, which is also shown for comparison. The output of the filter matched to the well-resolved cluster at velocity (-32.5,-58.9) μrad/sec is a near replica of the input frame for that cluster; tracks for these objects would therefore be initiated with the correct cluster velocity and acceleration. The remaining two filters (Figures 44c and 44d) completely pass the cluster to which they are matched, but it can be seen that some objects from the other cluster leak through due to marginal velocity resolution. As a practical matter, this means that the tracks for certain resolved objects in this threat would have two different velocity histories assigned to them at the initiation stage. This ambiguity would be resolved later in the track continuation process, or if desired, by processing additional scan frames in the velocity filters at initiation.

4.5.5 Bulk Discrimination. We have carried out preliminary experiments on the use of velocity filtering for bulk rejection of kinetic kill debris based on motion, using data provided by M.I.T. Lincoln Laboratory. For example, Figure 45 shows 6 data frames taken at 5 second intervals. Each frame represents a 120 x 120 wide field-of-view that has been "panned" to maintain the objects within the frame size shown. There are four main objects representing an RV and its associated decoys plus a debris cloud which expands rapidly to obscure the main objects in the later frames.

Since the main objects would generally be in track prior to the kinetic kill event, we assumed that their known track file velocity histories could be used to establish the set of four matched filters needed for track continuations. The frame sequence of Figure 45 was processed with each of these four filters to obtain the four output frames shown in Figure 46 (the output frames are referenced to the time of the sixth frame shown in Figure 45f). In each output frame the intensity of the particular object whose velocity lies within the passband of the filter is enhanced relative to that of the other main objects and the debris. Figure 47 compares the sixth input frames with a composite of the output frames in Figure 46 after thresholding. In this case, it is evident that the velocity filters have completely rejected the debris based on motion discriminants alone.

The motion discrimination capability of the velocity filter could be combined with other discriminants based on object intensity fluctuation or spectral content. One potential advantage of the motion discriminant is that it can be implemented via bulk signal processing with a bank of velocity filters operating in parallel. In such a configuration, the velocity filter bank could serve as the pre-processor in an overall discrimination architecture.

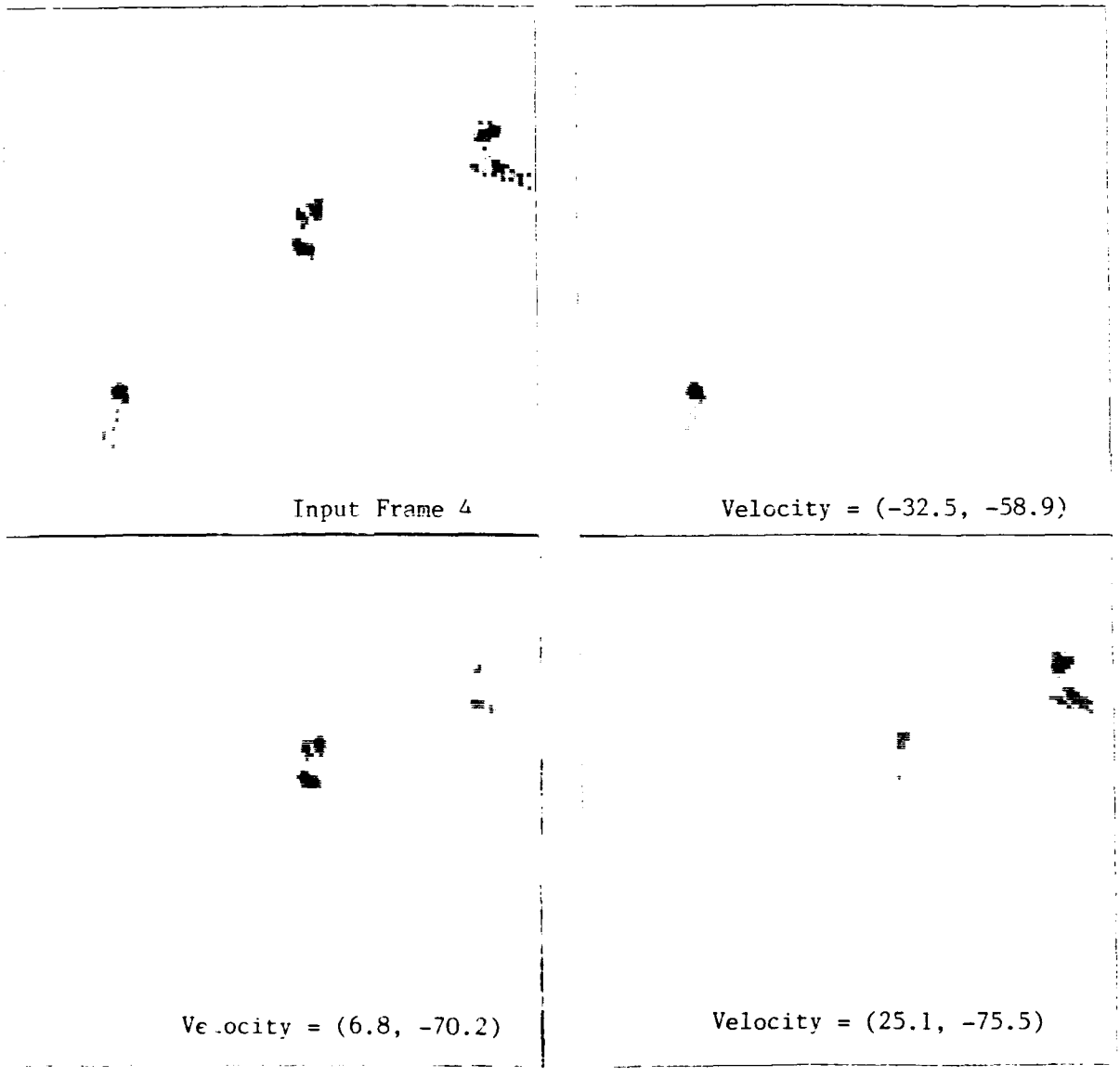


Figure 44. Input Frame Four, and Three Velocity Filter Outputs.

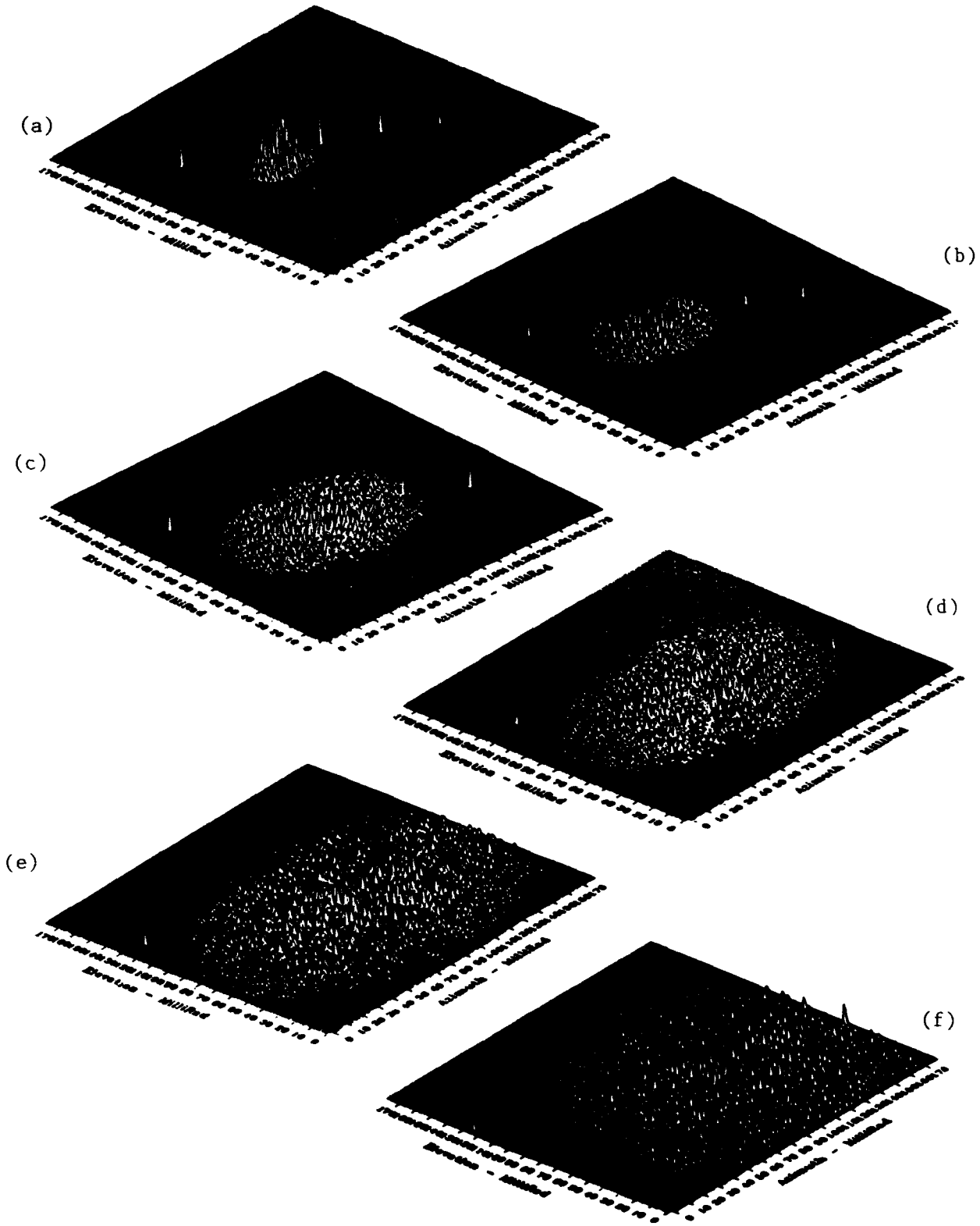


Figure 45. Six Images with Four Primary Objects Plus Debris.

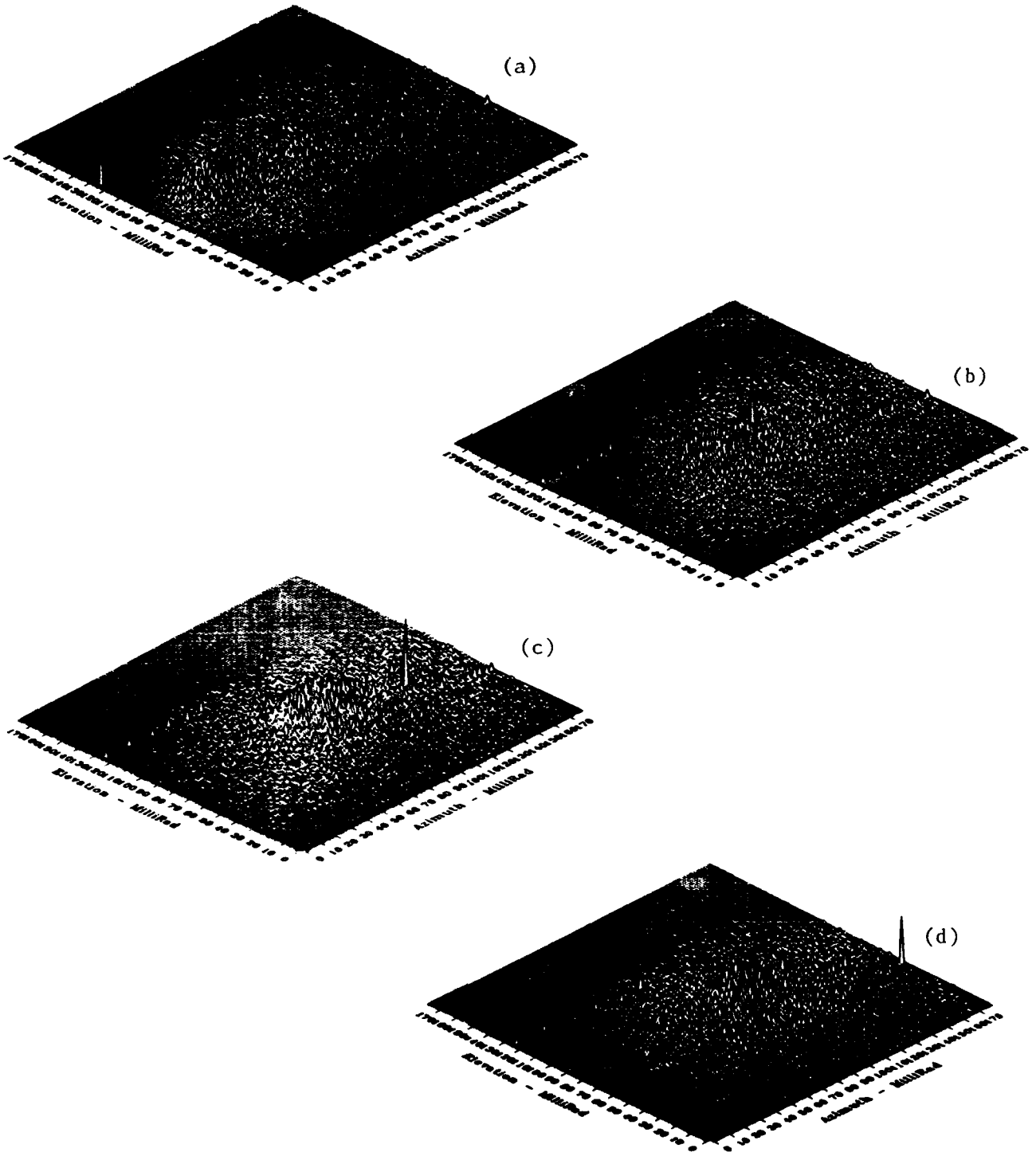


Figure 46. Output of Four Velocity Filters.

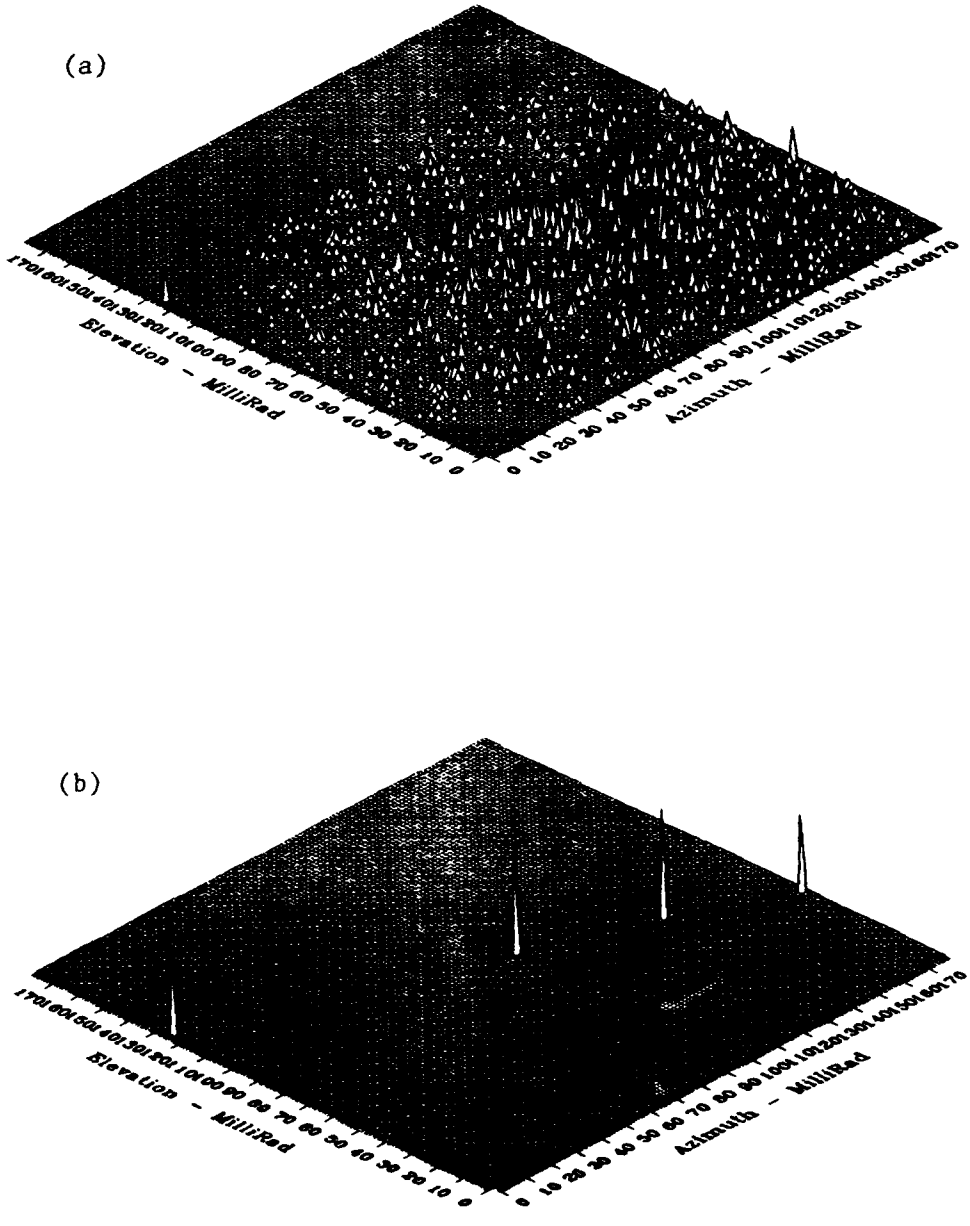


Figure 47. Comparison of (a) Sixth Input Frame and (b) Composite Output of Velocity Filters.

4.6 References

[1] C.H. Knapp and G.C. Carter, "The Generalized Correlation Method for Estimation of Time Delay," *IEEE Transactions on Acoustics, Speech, and Signal Processing*, Vol. ASSP-24, No. 4, pp. 320-327, August 1976.

[2] A.V. Oppenheim and R.W. Shaefer, *Digital Signal Processing*, Chapter 11, Prentice-Hall, 1975.

[3] R.E. Boucher and J.C. Hassab, "Analysis of Discrete Implementation of Generalized Cross-Correlator," *IEEE Transactions on Acoustics, Speech, and Signal Processing*, Vol. ASSP-29, No. 3, pp. 609-611, June 1981.

[4] I.S. Reed, R.M. Gagliardi and H.M. Shao, "Application of Three-Dimensional Filtering to Moving Target Detection," *IEEE Trans. on Aerospace and Electronic Systems*, Vol. AES-19., No. 6 (November 1983).

[5] W.J. Jacobi and L.A. Wadsworth, "Mineaturized Real-Time Processor for Image Sequence Analysis," *SPIE OE/Aerospace Sensing Symposium*, Orlando, FL (April 1990).

[6[P.L. Chu, "Efficient Detection of Small Moving Objects,", MIT Lincoln Laboratory Technical Report No. 846, 21 July 1989.

[7] Y. Barniv, "Dynamic Programming Solution for Detecting Dim Moving Targets," *IEEE Trans. on Aerospace and Electronic Systems*, Vol. AES-21, No. 1 (January 1985).

[8] M.F. Fernandez, A. Aridgides and D. Bray, "Detecting and Tracking Low-Observable Targets Using IR," *SPIE Proceedings*, Vol. 1305 (April 1990).

[9] J.F. Arnold and H. Pasternak, "Detection and Tracking of Low-Observable Targets Through Dynamic Programming," *SPIE Proceedings*, Vol. 1305 (April 1990).

[10] P.L. Chu, "Optimum Projection for Multidimensional Signal Detection," *IEEE Trans. Acoustics, Speech and Signal Processing*, Vol. 36, No. 5 (May 1988).

[11] S.D. Blostein and T.S. Huang, "Detection of Small, Moving Objects in Image Sequences Using Multistage Hypothesis Testing," *Proceedings ICASSP 1988*, Vol. 2 (April 1988).

[12] A. Wald, *Sequential Analysis*, Wiley, New York (1947).

[13] S. Tantaratana and H.V. Poor, "Asymptotic Efficiencies of Truncated Sequential Tests," *IEEE Trans. on Information Theory*, Vol. IT-28, No. 6 (November 1982).

[14] J.V. DiFranco and W.L. Rubin, *Radar Detection*, Artech House, Dedham, MA (1980).

[15] P.L. Chu, "Efficient Detection of Small Moving Objects," MIT Lincoln Laboratory Report 846 (July 1989).

[16] K. Preston, Jr., "Detection of Weak Sub-Pixel Targets Using Cellular Automata," *IEEE Trans. on Aerospace and Electronic Systems*, Vol. 26, No. 3 (May 1990).

[17] Drummond, O.E., S.S. Blackman and K.C. Hell, "Multipie Sensor Tracking of Clusters and Extended Objects," *Proceedings of the 1988 Tri-Service Data Fusion Symposium*, Laurel MD, 1988.

[18] Tsai, M., L.C. Youens and K. Dunn, "Track Initiation in a Dense Target Environment Using Multiple Sensors," *Proceedings of SPIE Conference on Digital Signal Processing, Association and Tracking of Point Source, Small and Cluster Targets*, Orlando FL, March 1989.

5.0 BRASSBOARD SIGNAL PROCESSOR

5.1 Computational Requirements

Real-time implementation of the generalized velocity-filter process shown in Figure 1 will challenge the throughput and memory capacities of the signal and data processing subsystems of IR surveillance systems. This is due not only to the larger volumes of data associated with multi-frame processing, but also to the increased computation required for algorithms which can optimally utilize multi-frame data.

A generic sensor processing chain is shown in Figure 48. Generally, the most stressing processing requirements occur at the so-called signal processing level, where the critical time-dependent functions such as clutter suppression, detection processing and early discrimination (based on pre-thresholded data) take place. Typical throughput requirements for these signal processing functions are well beyond currently-available general-purpose computers. For example, we have estimated the real-time processing load for a hypothetical staring sensor which generates data at a rate of one million pixels per second. Figure 49 shows throughput estimates by function for a digital signal processor which implements the algorithms required to extract weak point targets in clutter. The combined throughput for all algorithms in the processing chain is on the order of 2.5 GOPS.

Although the processing load for these functions is formidable, it is dominated by operations such as 2-D FFTs, array multiplications, discrete convolutions, and array shift-and-adds. These operations have very regular computational structures; i.e., they require little or no data-dependent branching. The computational problems that arise in IR signal processing therefore lend themselves naturally to pipelined and/or parallel processing architectures.

5.2 Processor Architectures

There are two fundamental approaches to digital processor design. One approach is based on the use of special-purpose hardwired circuitry, which is characterized by high speed but low flexibility. This approach has been adopted in the past for sensor signal processors because of high throughput requirements and because the algorithms utilized have been relatively simple and somewhat resistant to change.

The second approach is to employ a programmable processor. This approach has traditionally been employed for data processors, but in the future it will be increasingly applied to signal processing as the required algorithms become more complex, and as new semiconductor devices are developed to support parallel configurations of programmable microprocessors.

A wide variety of parallel processor architectures have been developed over the past decade or so. At one extreme, "coarse-grained" machines like the Cray X-MP use a small number of very powerful processors. Each processor has a sophisticated architecture and is built using the fastest available circuit technology. This approach, however, is unsuitable for most sensor applications because of the large number of devices, the large size and weight, and the high levels of power required.

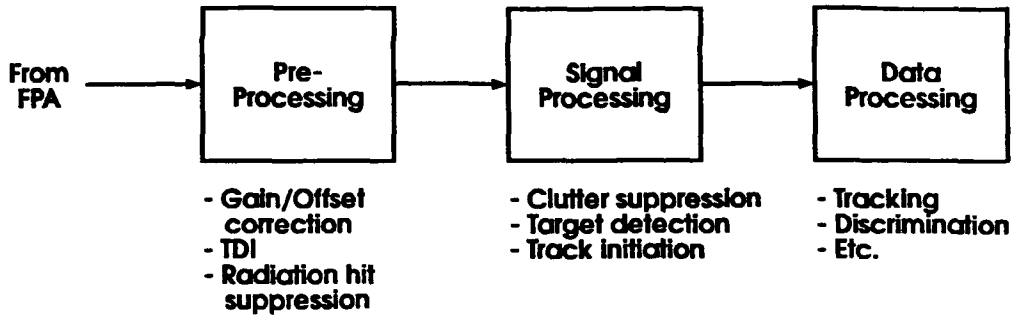
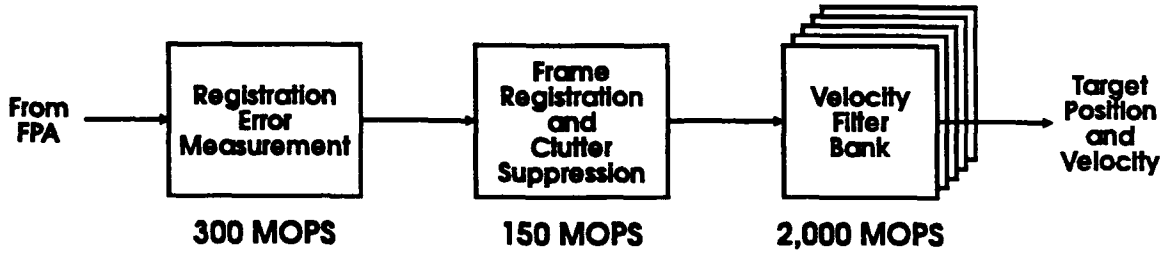


Figure 48. Generic Sensor Processing Configuration.



Assumptions:

Frame Size = 1K x 1K pixels
Inter-Frame Period = 1 second
Velocity Filter Length = 64 frames
Number of Velocity Filters = 1,000

Figure 49. Throughput Requirements for Hypothetical Sensor.

At the opposite extreme are the "fine-grained" or massively-parallel machines based on very simple, bit-serial processing elements. Although any one processor is of little value by itself, the aggregate computing power can be very large when many such processors are coupled together. A well-known example of this approach is the Connection Machine, which uses tens of thousands of one-bit processors to achieve throughputs in excess of 2,500 MIPS. Other such machines include the DAP 510 Array Processor (developed by Active Memory Technology, Inc.) and the Associative String Processor (developed by Brunel University and Aspex Microsystems, Ltd.). Although these fine-grained architectures based on bit-serial processors can be extremely fast in terms of raw computational power, the input/output requirements tend to limit their utility in real-time signal processing applications.

A third approach is based on combining an intermediate number (tens to hundreds) of medium-grained processors. Among the many possible architectures in this category is the so-called multicomputer network. Each of the identical nodes in the network contains a microprocessor with local memory and provisions for communicating with other nodes. In general, there is no central shared memory; the cooperating tasks of a parallel algorithm execute asynchronously on different nodes. Current examples of the medium-grained approach include the Caltech Hypercube and its descendants, and the Carnegie-Mellon/Intel iWarp.

The medium-grained approach is the basis for many of the systolic array architectures which have proven to be useful for a number of matrix arithmetic and signal processing applications, including matrix inversion, eigenvector-eigenvalue decomposition, digital filtering and correlation, the FFT, and sorting and thresholding problems. Such operations will clearly be required in any general-purpose, multi-spectral IR signal processor. These algorithms share the key attributes of regularity, recursiveness and local communication. These attributes can be effectively exploited in multicomputer networks, particularly those which have a pipelined, data-driven configuration, and it seems likely that variations of the medium-grained approach will be increasingly applied to sensor signal processing tasks. However, for the shift-and-add part of the velocity filter bank it is still attractive to use special-purpose hardware with only limited programmability (see section 4.4.3).

5.3 Processor Design

In accordance with the foregoing considerations, Space Computer Corporation has developed a medium-grained, high-throughput, programmable parallel processor. This processor, called the SCC-100, is specifically designed for real-time signal processing applications utilizing large data sets such as those derived from IR sensors. It utilizes a multiple-instruction, multiple-data (MIMD) architecture, with adjacent processing nodes interconnected through banks of shared memory. Each node has a peak throughput of 100 MFLOPS using 32-bit IEEE-standard floating-point arithmetic, so that a system of 36 nodes has a peak throughput of 3.6 GFLOPS.

5.3.1 Architecture. The system architecture of the SCC-100, shown in simplified form in Figure 50, consists of a linear array of high-performance processing elements (labeled PE). Each of these elements shares dual-banked memory with its adjacent elements in the array. The banks (labeled M) are switched from one processing element to the next in a single memory cycle, thus providing extremely high-speed

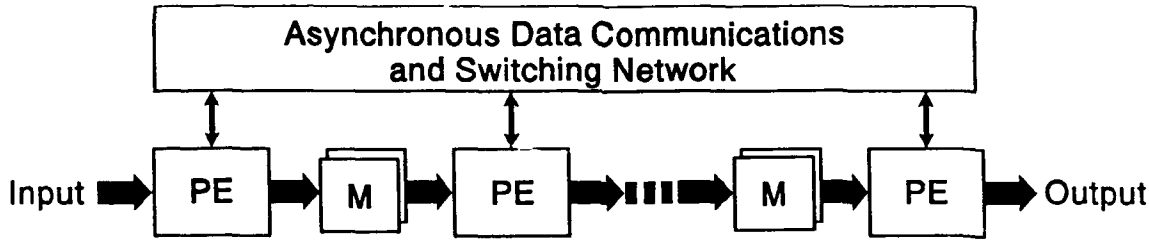


Figure 50. Simplified SCC-100 Architecture.

data transfer for large volumes of data. (If this were an actual channel, the equivalent bandwidth between adjacent nodes would be 4×10^4 MBytes per second.) A related benefit of the dual-banked memory is that it enables data transfers between nodes to be made without program overhead, thereby increasing efficiency. For problems which map naturally onto the pipeline model (such as almost all signal and image processing problems), the total throughput for N nodes approaches N times the throughput for a single node. This is significantly better than the $\log_2 N$ speedup usually approached with some other architectures.

The block diagram of a single processing node and memory bank is shown in more detail in Figure 51. As can be seen from that figure, each processing element has four bi-directional, asynchronous, high-speed (20 MBits/second) serial links for data communications and control. Two of these links are connected to the adjacent elements in the array, while the other two links are connected to a crossbar switching system which can connect any element to any other element. In this manner arbitrary network topologies can be established for efficient algorithm mapping. All of the control and switching operations are implemented in hardware for high speed with low overhead.

Each of the processing elements contains a 20-MIPS (peak), 32-bit RISC microprocessor (INMOS T800 Transputer), and up to three 32-bit vector signal co-processors (Zoran ZR34325), in addition to one MByte of local memory, and one MByte of dual-banked memory. The T800 Transputer incorporates a number of important features which make it ideal for use in real-time parallel computer systems. These include four high-speed asynchronous serial communications links, 4 KBytes of on-chip RAM, two 32-bit timers, hardware support for multi-tasking, and a 32-bit IEEE-standard floating-point co-processor (3 MFLOPS peak throughput), all on a single device.

The ZR34325 Vector Signal Processor, which also performs 32-bit IEEE-standard floating-point arithmetic, has a peak throughput of 33 MFLOPS (a total of 99 MFLOPS for three units). This powerful chip has a high-level instruction set optimized for commonly-used signal and image processing operations. A complex FFT operation, for example, requires only a single instruction. This feature greatly simplifies the programming required to implement digital signal processing algorithms. With three ZR34325s, a node can compute a 1024-point complex FFT in 0.72 milliseconds.

Two or more of the nodes can have high-speed input/output interfaces for data acquisition and display. In addition, the SCC-100 is operated with an AT-compatible PC host computer. The host is used for software development, program downloading, and performance monitoring. For a production system a host is not required. In that case the host can be replaced by a set of ROMs.

5.3.2 Input/Output. High-speed data communications channels are available with the SCC-100 for connection of the processor to external data sources and destinations. Each of these channels is capable of transferring data at a rate of 40 MBytes/second. In addition, multiple high-speed control channels (each with a bandwidth of 2.4 MBytes/ second) are provided. Additional I/O channels can be furnished for access to specific nodes along the array. Also, custom I/O channels to meet special interface requirements can be provided.

A critical problem for many real-time processor applications is real-time input data acquisition. This requires a capability for servicing interrupts at a rapid rate to maintain high-fidelity signal capture.

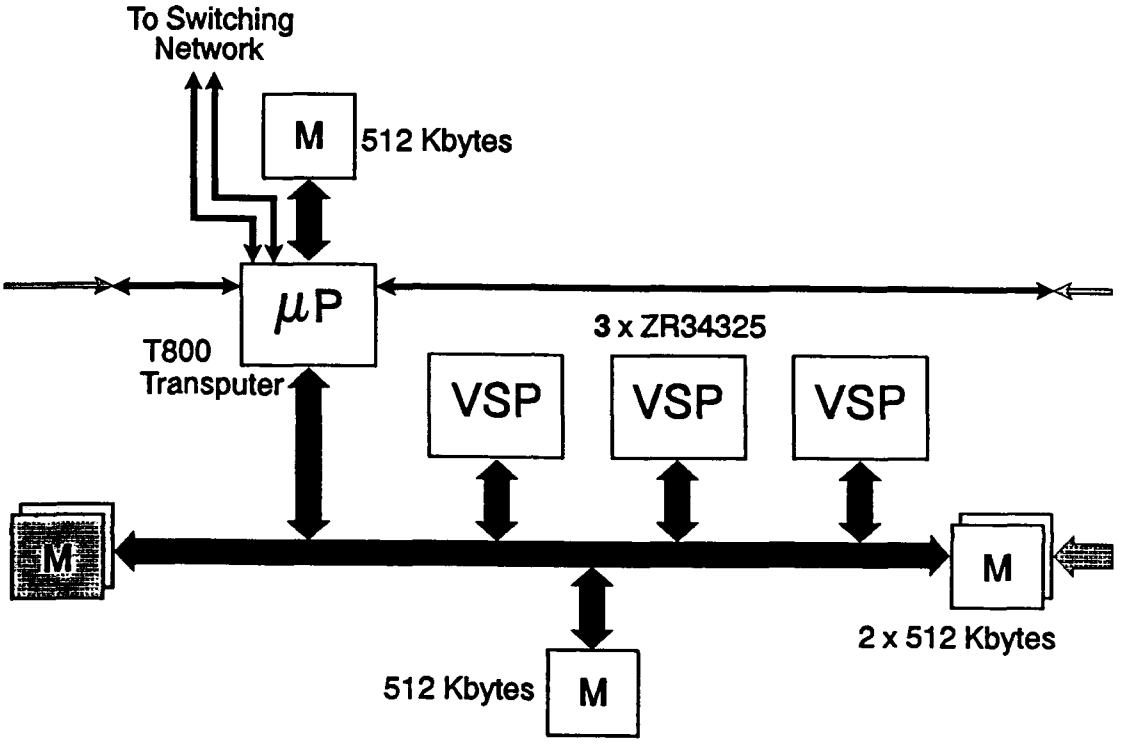


Figure 51. Block Diagram of Single Processing Element and Memory Bank.

In the SCC-100, over 400,000 interrupts per second can be serviced in a single input node. If necessary, this capability can be increased further by utilizing additional processors.

5.3.3 Programming. The fundamental programming task in the SCC-100, as in any parallel processor, is program partitioning. There are three basic partitioning methods:

(a) Input partitioning (e.g., for low-level image processing);
(b) Output partitioning (e.g., for irregular input-to-output mapping);
(c) Pipelining.

The SCC-100 supports all three of these methods. As previously noted, however, it is particularly effective for pipelining, which is by far the most important and effective partitioning method for most signal and image processing algorithm chains. In pipelining, the algorithms rather than the data are partitioned. This greatly eases the programmer's task, since the program for each node can usually be written and tested independently of the other nodes prior to final software integration.

The SCC-100 processor can be programmed in a variety of commonly-used languages. The T800 Transputers can be programmed in ANSI C and several other languages. The Zoran Vector Signal Processors can be programmed using a library of routines produced by Space Computer Corporation. In addition, assembler/simulator software, which uses C syntax and which runs on a PC/AT or VAX/VMS host computer, is available for the Zoran processors. In addition to this, both processors can be programmed in ADA using compilers available from Alsys and PSS.

5.4 Advanced Packaging

The standard model of the SCC-100 is packaged with conventional multilayer printed-circuit boards. Models for applications requiring extremely compact size and ultra-low weight are being fabricated using hybrid wafer-scale integration (HWSI). Figure 52 shows the planned evolution, starting with the initial model which houses 20 nodes, one node per board, in a 14"-high cabinet which can be mounted in a standard 19-inch rack. The first step (1990) is replacement of a large number of glue chips by two application-specific integrated circuits (ASICs) in each node. This reduces the size so that two nodes can be packaged on a single board, and reduces the chip count so that HWSI packaging can be employed.

The next step (1991) applies 2-D HWSI technology to package each node as a multi-chip module rather than as a printed-circuit board. The HWSI modules employ bare chips attached to thin-film, multilayer, polyimide interconnects fabricated on silicon substrates within hermetically-sealed packages. The size of each module is approximately 3.0 x 3.5 x 0.2 inches. These modules in turn are mounted on printed-circuit boards. Twenty nodes in this configuration have a volume of approximately 200 cubic inches and a weight of 8 pounds. The power consumption will be reduced from that of the 1990 configuration by virtue of the shorter dimensions and lower dielectric constant of the HWSI interconnects, which provide greatly reduced line capacitances.

The final step (1992) utilizes stacked HWSI substrates in a 3-D configuration corresponding to a cube 2.5 x 2.8 x 3.5 inches in size. A twenty-node system in this configuration will have a volume of 25 cubic inches and a weight of 2 pounds, including package and connectors.

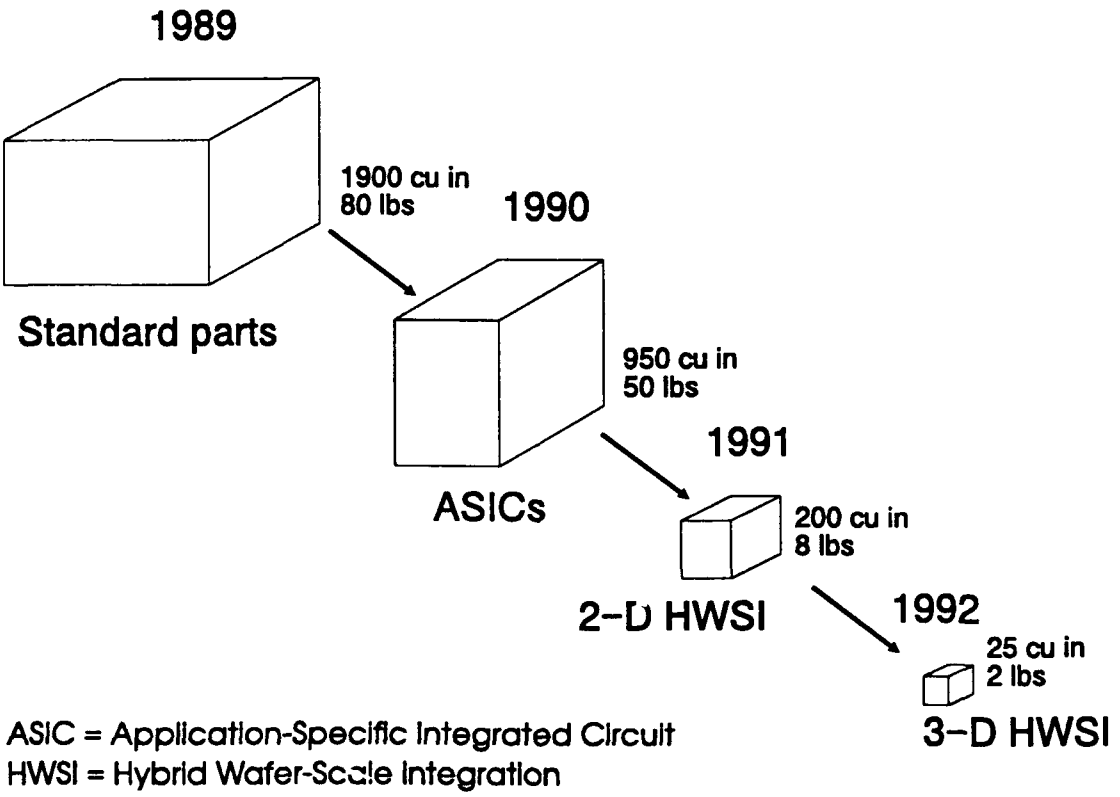


Figure 52. SCC-100 Processor Packaging Evolution for a 2-GFLOPS, 40-MByte System.

6.0 SOFTWARE FOR REAL-TIME DEMONSTRATION

6.1 General

Real-time operation of the SCC-100 has been demonstrated using input frames of size 128 x 128 16-bit pixels at a rate of 10 frames per second. Thus, the input pixel rate is 164k pixels per second, or 328k bytes per second, or 2.62 Mbits per second. The functions implemented for real-time processing are frame registration, temporal clutter suppression, a bank of 80 discrete shift-and-add velocity filters, and thresholding for target detection. These signal processing functions are performed sequentially on successive batches of input frames to periodically produce sets of detection reports. The target aquisition signal processing chain is shown in Figure 53. Each output "detection" consists of the 2-D apparent position and velocity of an object on the focal plane. The velocity filter detection reports are the primary output of the real-time signal processor.

The demonstration setup is shown in Figure 54. Input can be taken from either a video camera or an optical video recorder. The image frames from the selected device are fed to a frame grabber housed in a PC for digitization into 16-bit samples. Though these devices produce frames at a rate of 30 frames per second, only every third frame is actually used, so the input frame rate is actually 10 frames per second. Also, only a selected 128 x 128 subframe from the frame grabber is actually fed to the SCC-100. As indicated in the figure, one PC serves as the SCC-100 host, and handles program downloading and operator intervention. Finally, a second frame grabber is connected to a 16-bit parallel output from the SCC-100 for the display of images at intermediate points in the process.

When each frame arrives at the first processing node, it is tagged with a number which indicates the time the frame was collected, and the 16-bit pixel values are converted to 32-bit IEEE floating point format. All subsequent processing is performed in floating-point format.

Target aquisition processing is performed on consecutive groups of frames called processing batches. The number of frames in a processing batch is a basic software parameter which specifies (a) how often the reference frame for frame registration is updated to be the current frame, and (b) the number of frames that are co-registered, clutter suppressed, and integrated in the velocity filters. It also determines the time interval at which detections are produced. For the demonstration the batch size was 10 frames. Thus, a total of 10 frames were integrated in each velocity filter, and a new set of output detections was generated every 10 frames (i.e., once per second).

The frames in each processing batch batch are spatially registered to the first frame of the batch as they are received. At the beginning of each batch the misregistration between the new reference frame and the previous reference frame is computed and output along with the velocity filter detections from the new batch. This allows the detections for the new batch to be spatially referenced to those for previous batches.

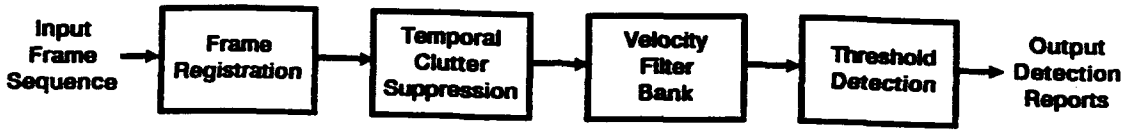


Figure 53. Target Acquisition Signal Processing Chain.

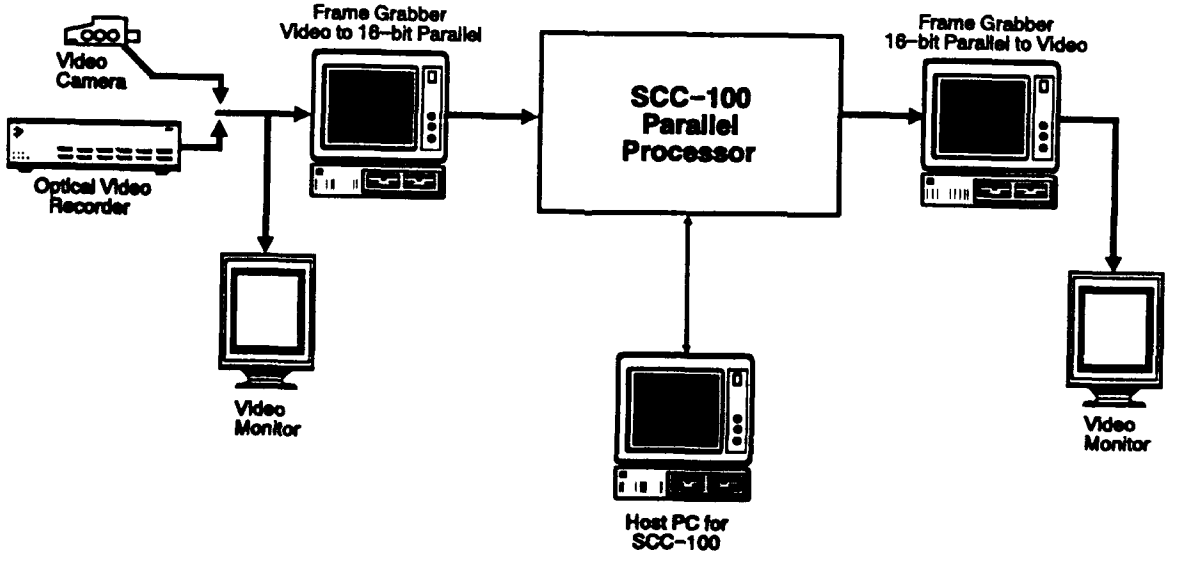


Figure 54. Demonstration Setup Using Frame Grabbers in PCs for Frame Capture and Display.

6.2 Allocation to Nodes

The real-time demonstration processor is a 19-node system, and each of these nodes is populated with just two Zoran VSPs, rather than the standard three VSPs. Sixteen of the nodes are dedicated to performing calculations, two nodes are used to perform calculations and to handle input/output functions, and one node is used to control the system and communicate with the host computer. This allocation was established empirically by first coding the actual subfunctions to be performed and timing each one of them. (The Transputer contains hardware which makes it very easy to obtain accurate timing for any code executing either in itself or in the VSPs.) After the timing was obtained, the subfunctions were assigned to nodes in such a way as to avoid bottlenecks, and to keep all nodes working as efficiently as possible. The effectiveness of this approach to partitioning the software among nodes was verified by the fact that it was not necessary to reassign the subfunctions to nodes after this initial straightforward assignment procedure was followed.

The first node (node number 0) controls the system, communicates with the host, and reports the maximum value in the output of each of the velocity filters over a serial port. This node also monitors and controls the communications through the processor pipeline.

Nine nodes (node numbers 1 through 9) are used for frame input and data formatting, and for carrying out the computations required by the two parts of the frame-registration algorithm, viz., measurement of the amount of 2-D shift required, and resampling the frame with the measured shift applied. Figure 55 shows the allocation of nodes to the various frame-registration functions. In that figure the labels in the blocks refer to subroutine names. The functions of those subroutines are shown below the blocks.

The next eight nodes are used as two sets of four nodes in a "flip-flop" fashion. While one set of four nodes gathers a processing batch of 10 frames and calculates the mean and variance for each pixel, the other set of four nodes applies clutter suppression to each pixel in each frame (mean subtraction and multiplication by the reciprocal of the standard deviation), and then shifts and adds its set of 10 clutter-suppressed frames to perform velocity filtering. This shift-and-add algorithm is repeated for 80 different 2-D shifts, thus implementing 80 different velocity filters. After one second of this processing, the first set of four nodes is ready to clutter suppress and velocity filter its frames, and the second set of four nodes is ready to start collecting the next 10 frames of data from the frame-registration nodes. It should be noted here that with this arrangement the four nodes which are computing the means and variances are not using their full computational capacity, but this utilization of nodes was found to be effective overall.

The last node is dedicated to collecting and formatting data for the output display. This function would not be required in an operational system, but is very useful for a demonstration system. See section 6.4 for more details on this function.

6.3 Processor Functions

Figure 53 shows the high-level block diagram of the processing sequence. Functional descriptions for each major processing task are as follows.

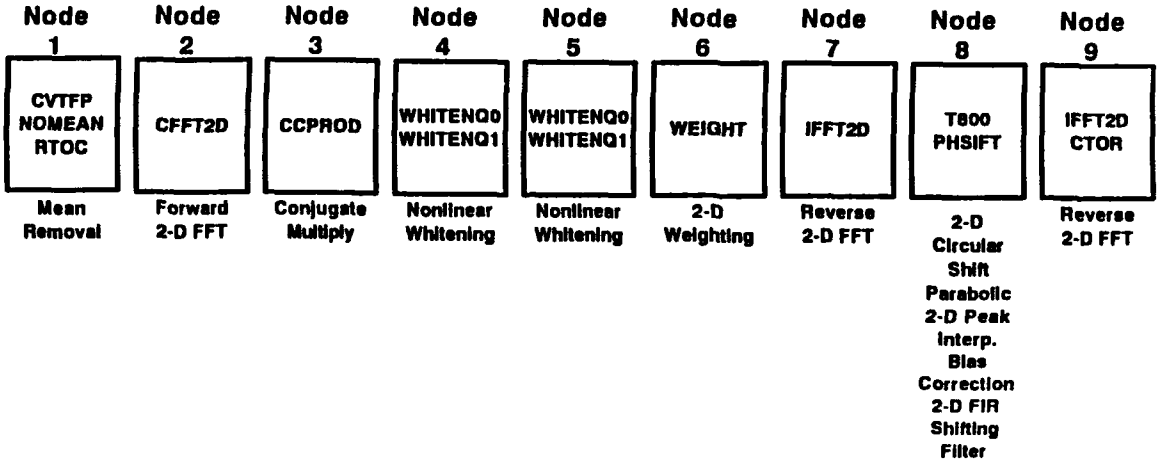


Figure 55. Frame-Registration Allocation to Processor Nodes.

6.3.1 Frame Registration. The frame registration algorithm corrects for 2-D translational jitter or drift between the frames of a processing batch, and consists of two steps: (1) measurement of the apparent motion of the clutter background with respect to the reference frame, and (2) resampling of the input frame to produce a new frame in which the background is registered with that in the reference frame. The objective is to achieve a precise spatial alignment of the background in all frames of the batch. This ensures that subsequent clutter suppression processing will be fully effective. The frame registration algorithm employed here utilizes the phase correlation technique described in section 4.2 to measure the 2-D misregistration to subpixel accuracy, and a FIR interpolation filter for high-precision resampling of the input frame.

A functional block diagram of the complete frame registration process is shown in Figure 56. The processing data flow is almost entirely pipelined with the single exception of the 2-D transform of the current input frame being required at two different points in the chain (for correlation and also for frame shifting). Every 2-D frame operation shown in the block diagram (i.e., FFT, weighting, and multiplication) is performed on a full 128 x 128 real or complex-valued array. The discrete correlations and frame shifts are implemented in a circular fashion using the full 128 x 128 frames without zerofill.

For each processing batch, the first frame is used as the reference for registration measurements. The 2-D FFT of the reference frame, denoted as "Stored Reference Frame Transform" in Figure 56, is retained for conjugate multiplication with the transforms of subsequent frames as they are computed. Thus all misregistration measurements are made with respect to this reference frame, and each frame in a batch is shifted so that it is aligned with its reference frame.

When the first frame of a new processing batch arrives, the reference frame is updated, but only after the new reference frame is correlated against the previous reference frame to measure the 2-D offset between successive reference frames. This information is produced for every batch, and is transferred to the system output node so that the positions of detections made in the new processing batch can be related to those made in the previous batch. The 2-D FFT of the new reference frame then replaces the old reference frame transform prior to processing additional frames from the new batch. Since the FIR shifting filter applies spatial filtering in addition to linear phase shifting in each dimension, each reference frame must be similarly filtered (but with zero shift) to maintain the same spatial frequencies as the other frames in the batch.

6.3.2 Clutter Suppression. Clutter suppression is a frame whitening process which suppresses the background in a batch of frames prior to velocity filtering. The input is the sequence of spatially-aligned frames that emerge from the frame registration process. The output is another frame sequence in which the stationary background is suppressed and normalized on a pixel-by-pixel basis. For every frame, the suppression process subtracts the spatially-varying mean background from each pixel, then normalizes the residual clutter in each pixel to unit power by dividing by the pixel standard deviation. (See section 4.5.3 for a discussion of the reasons for using this procedure.)

The estimation of the means and variances is implemented recursively on the registered frames in the processing batch as they become available. This results in the creation of a pixel mean array and a pixel variance array, each of size 128 x 128. The recursive filtering scheme that is used to update the pixels in these two arrays is illustrated in Figure 57. The mean and variance arrays are initialized to all zeros at

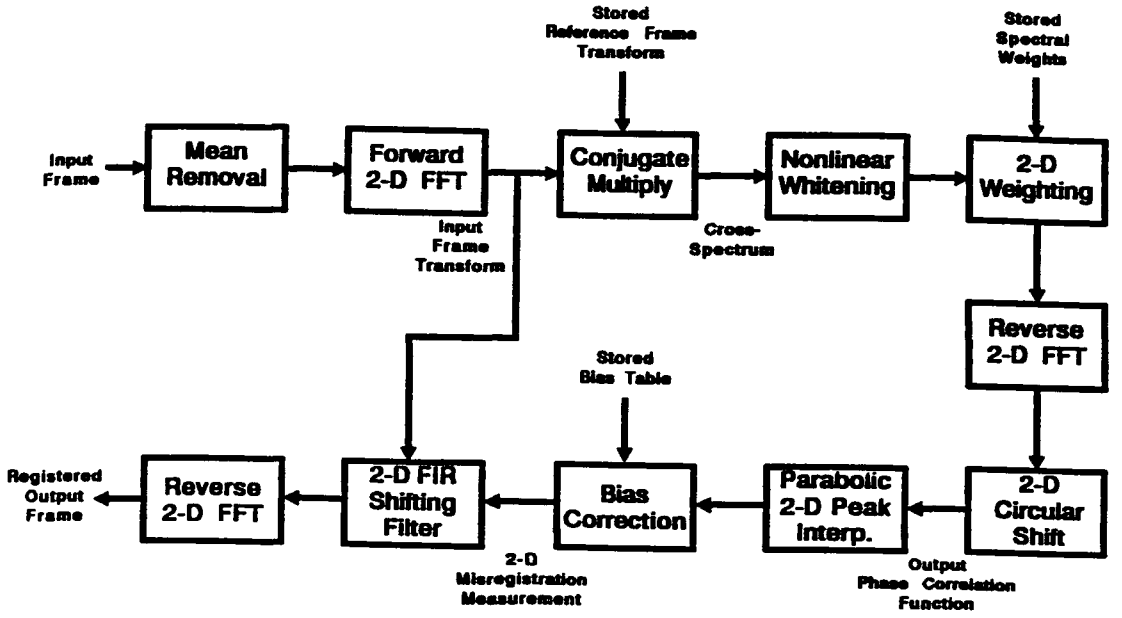
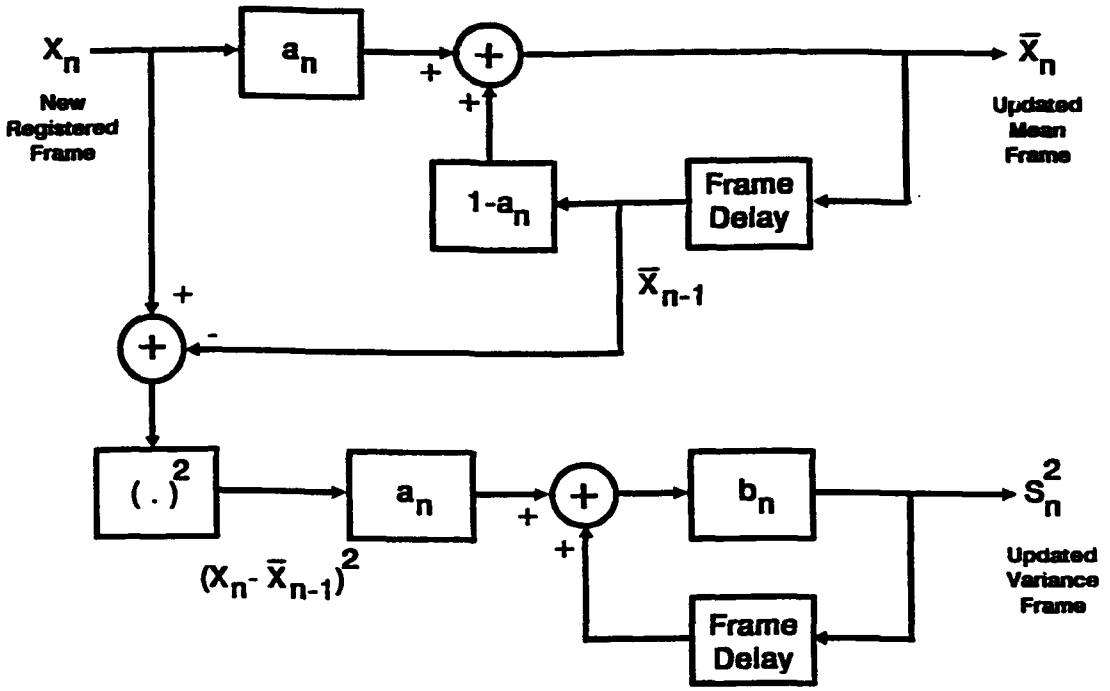


Figure 56. Block Diagram of Frame Registration Algorithm.



Update Equations:

$$\bar{X}_n = a_n X_n + (1-a_n)\bar{X}_{n-1} \quad \text{(mean)}$$

$$S_n^2 = a_n b_n (X_n - \bar{X}_{n-1})^2 + b_n S_{n-1}^2 \quad \text{(variance)}$$

Initial Conditions:

$$\bar{X}_0 = 0$$

$$S_0^2 = 0$$

Filter Coefficients:

$$a_n = 1/n$$

$$b_n = (n-1)/n$$

Frame Indices:

$$n = 1,2,\ldots,N_{batch}$$

Figure 57. Recursive Scheme for Estimating Mean Frame $\bar{x}_n$ and Variance Frame s_n^2.

the start of every processing batch. As successive registered frames in the batch are received, the arrays are updated according to the expressions shown in Figure 57. When all the frames in the batch have been processed, the arrays contain the final temporal mean and variance estimates for each pixel in the frame. The scalar filter coefficients a_n are precomputed and stored in a table. The particular choice of weights shown in Figure 57 ensures the the final mean and variance estimates for each pixel are equal to the standard sample mean and variance for the frames.

The normalization step implements the actual clutter suppression process on each frame in the batch. This commences immediately after the last registered frame in the batch has arrived and the final updates of the mean and variance arrays have been completed. A block diagram of clutter suppression processing is shown in Figure 58. The normalization weights for each pixel are first obtained by converting the final variance array to reciprocal standard deviation. This requires square root and reciprocal operations for each variance-array pixel. The clutter suppression is performed on each registered frame in the batch by subtracting the mean frame and multiplying the result by the reciprocal standard deviation frame. These operations are done in-place on each registered frame in the batch. The output of the clutter suppression process is a sequence of whitened frames ready for velocity filtering.

6.3.3 Velocity Filtering. Velocity filtering is the frame integration process which enhances the SCR of targets moving through the whitened frame sequence. A specific velocity filter shifts and adds successive normalized frames in the processing batch to create a single output frame in which the responses of any targets moving at (or near to) the 2-D velocity of the filter are enhanced relative to the whitened clutter. Since targets of interest can have any of a wide range of 2-D velocities, it is necessary to implement a "bank" of velocity filters. Each filter in the bank is sensitive to a particular 2-D velocity, which is specified by means of a shift table. The velocity filter bank is illustrated in Figure 59.

Shift-and-add processing for a particular velocity filter is completely specified by the filter velocity and the number of frames to be integrated. Specifically, the 2-D shift applied by a filter to a given frame in the batch is a function of the frame number, the total number of frames processed, and the normalized 2-D velocity to which the filter is matched (in units of pixels per frame). Successive frames in the batch are shifted ahead in time so that target positions in the filter output frame will be time-referenced to the time of the last frame in the batch. The 2-D shifting is performed in a non-circular fashion; pixels which would be shifted outside the original frame boundaries are discarded. Note that a positive shift corresponds to the direction of increasing pixel index in either frame dimension (i.e., upward or rightward translation of a frame with pixel (0,0) at the lower left corner).

The shift-and-add integration scheme that implements a single velocity filter output is illustrated in Figure 60. Note that although the 2-D filter velocity can be a fractional number of pixels per frame, the actual shifts applied to any frame are rounded to the nearest whole pixel in order to simplify the real-time processing. The sequence of 2-D integer pixel shifts associated with a particular filter are precomputed off-line and stored in a shift table for downloading at program initialization. Each velocity filter in the bank has a unique shift table corresponding to its assigned 2-D velocity.

6.3.4 Detection Thresholding. Each pixel in each velocity filter's output frame is compared to a threshold to determine the positions of targets in the processed frames. The positions and the filter's 2-D

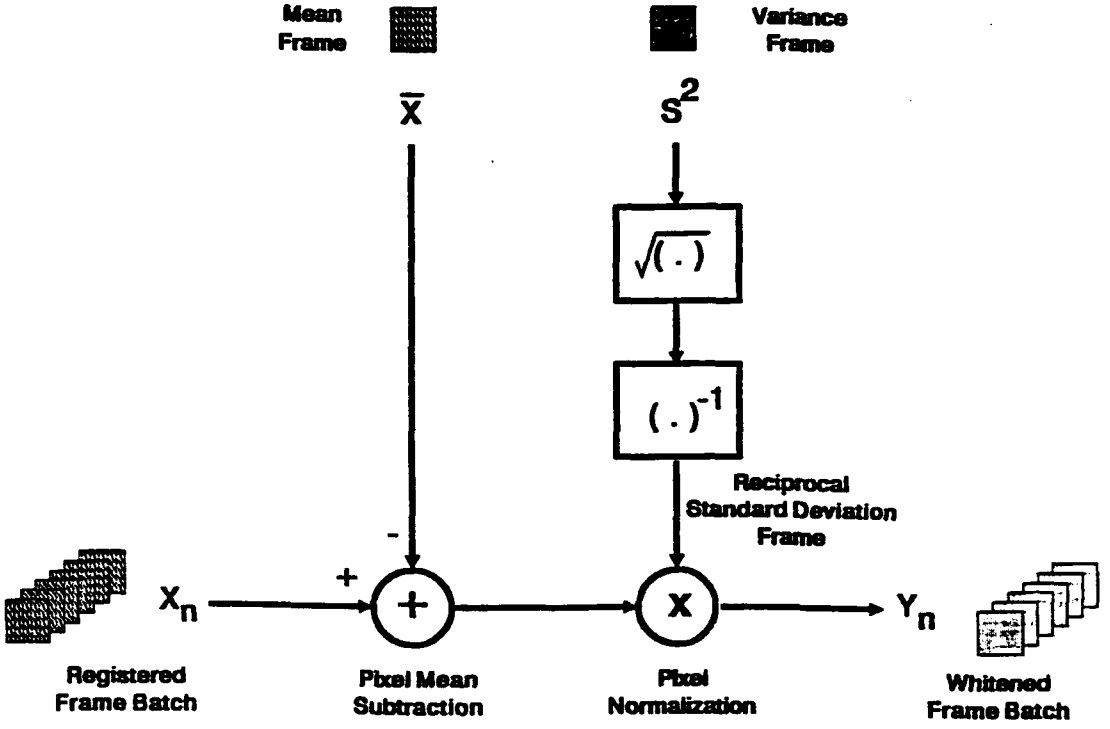


Figure 58. Temporal Clutter Suppression Processing.

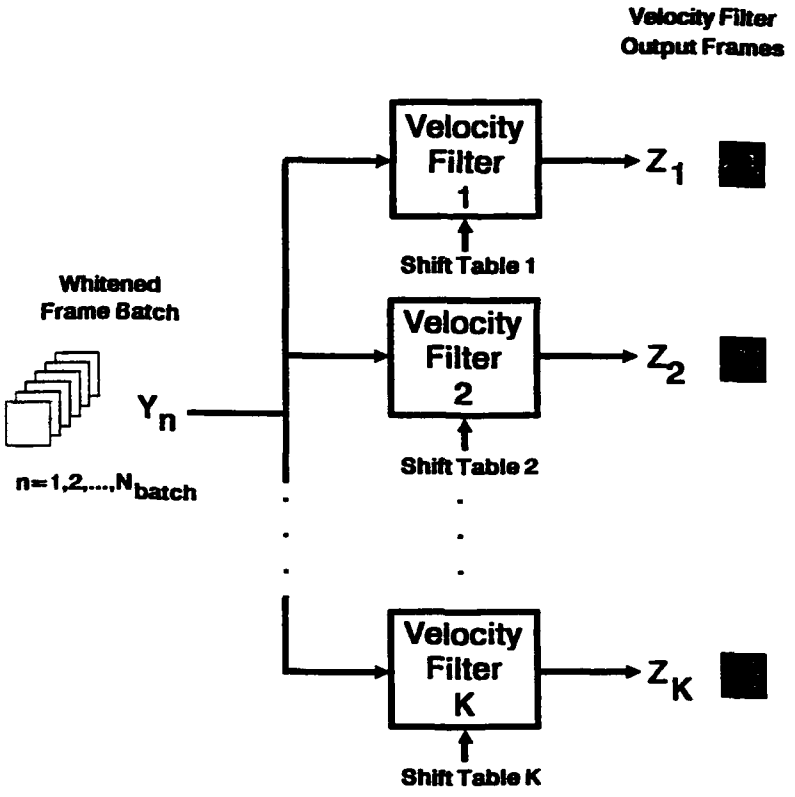


Figure 59. Velocity Filter Bank Concept.

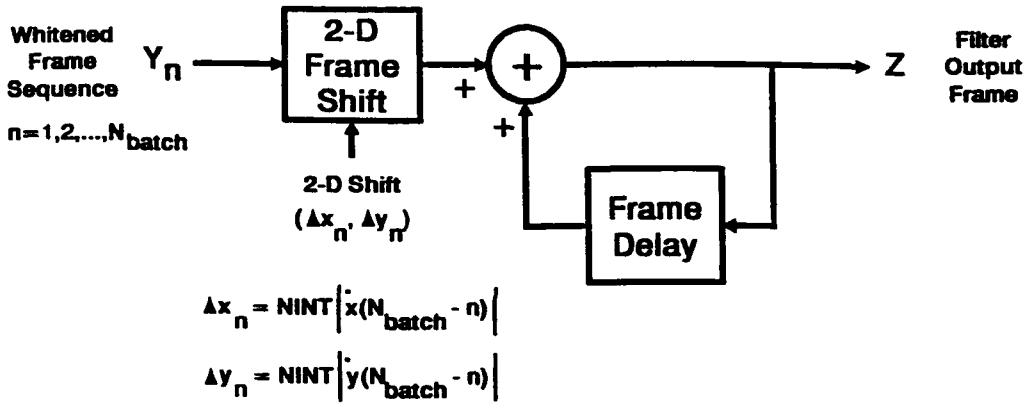


Figure 60. Block Diagram of Velocity Filter Processing.

velocity comprise the output detection reports. In addition, each velocity filter's output is searched to determine its maximum value, and the velocity filter with the maximum value out of all of the filters in the bank is output as part of the detection report for each processing batch.

6.4 Output Displays

The output of the SCC-100 required for the demonstration is just the detection reports, i.e., the positions and filter velocities of detected targets. However, it is much more interesting to see the results at the outputs of the various processing steps. Therefore, as shown in the demonstration setup of Figure 54, provision is made for 16-bit parallel transfers of data from the SCC-100 to an output frame grabber. The frame grabber converts the data received into NTSC video for display on a monitor. For the real-time demonstration, several different sets of output frames are available. The first of these is the frames which are produced by the frame registration algorithm. Next is the recursively-computed mean frame used in the clutter suppression. Following that is the recursively-computed variance frame used in the clutter suppression. And finally, the output of any one of the 80 velocity filters can be displayed.

Before outputting the selected data to the output frame grabber, the internal floating-point numbers of the SCC-100 must be converted to bytes which determine the intensity of a given pixel on the output display. This allows for just 256 possible intensities to be displayed for any pixel. Therefore, there are many different ways the floating-point values can be mapped onto the final display. The demonstration software allows this mapping to be specified as the program runs. The 256 intensities can be assigned linearly to the values between the maximum and minimum of each displayed frame, or fixed limits can be specified. This latter option allows for hard-limited or other special displays which may be desired.

7.0 CONCLUSIONS

The basic purpose of this research was to develop and demonstrate a new approach to the detection of, and initiation of track on, moving targets using data from a passive IR or visual sensor. The approach we have developed differs in very significant ways from the traditional approach of dividing the required processing into time-dependent, object-dependent, and data-dependent processing. Our approach is sometimes referred to as "track before detect" because the *detection* of targets is based on multiple image frames, and, accordingly, requires a smaller signal-to-noise ratio. This can lead to a significant reduction in total system cost, because it can allow greater detection range for a single sensor, or it can allow the use of smaller sensor optics. Both approaches are applicable to systems using scanning sensors, as well as those which use staring sensors. The advantages of our new approach can be summarized as:

a) It provides better detection for a given false alarm rate, and allows detection of very weak targets in clutter;
b) It is robust in that it is able to handle large numbers of targets simultaneously with no significant increase in computational resources;
c) It is readily implementable using a parallel processing architecture.

Under this program we have:

a) Developed and demonstrated the required algorithms;
b) Designed and fabricated a programmable real-time processor which can implement these and other signal and image-processing algorithms;
c) Programmed the algorithms to run in real-time on our processor, and demonstrated real-time operation using recorded imagery.

This research effort has had two major thrusts. The first is algorithm development, combined with proof-of-concept computer simulations using real and synthetic sensor data. The second is design and demonstration of a programmable, extremely high-performance signal processor which can be implemented with available components, and which can be packaged using advanced interconnect technology to achieve the small size and weight required for SDI and other military applications. The processor we developed employs a one-dimensional array of identical processing nodes. The peak throughput per node is 100 MFLOPS, and each node incorporates 2 MBytes of static RAM memory. Therefore, a 20-node system has a peak throughput of 2 GFLOPS and incorporates 40 MBytes of memory. In its basic form, using conventional printed circuit-board packaging, each node occupies a single board in a 14-inch high cabinet, and 20 such boards can be mounted in one 19-inch wide cabinet. When implemented with hybrid wafer-scale interconnect (HWSI), a 2-GFLOPS processor can be packaged in 35 cubic inches, including radiation shielding.

Military applications of our work include such programs such as Brilliant Eyes, Brilliant Pebbles, E^2I, EndoLEAP, and smart weapons. Commercial applications include video compression, computer vision, and some robotics applications.

It should be noted that this program has been very successful, not only from the technology viewpoint, but from the viewpoint of the goals of the SBIR program itself. For example, we have already proceeded into Phase III of this program, and were awarded a contract by General Dynamics Corporation for the delivery of a 19-node model of our processor. That first model was delivered to General Dynamics in early 1990. We are currently marketing the processor for both government and non-government applications. In addition, in 1990 we were awarded contract DACA76-90-C-0002 for DARPA (by the U. S. Army Engineer Topographic Labs) to miniaturize the processor for space applications. Finally, our work on moving-target detection has provided the basis for a new algorithm for full-motion video compression. Commercial applications for this new algorithm include high-definition television (HDTV), interactive video, and multimedia presentations.